AWS SDK for Java 2.x

Developer Guide for version 2.x

AWS SDK for Java 2.x: Developer Guide for version 2.x

Table of Contents

Developer Guide - AWS SDK for Java 2.x 1
- Get started with the SDK 1
- Develop mobile applications 1
- Maintenance and support for SDK major versions 1
- Additional resources 1
  - Features not yet in version 2 of the SDK 2
  - Contributing to the SDK 2

Get started tutorial 3
- Step 1: Set up for this tutorial 3
- Step 2: Create the project 3
- Step 3: Write the code 6
- Step 4: Build and run the application 9
  - Success 10
  - Cleanup 10
- Next steps 10

Setup 11
- Basic setup 12
  - Overview 12
  - Sign-in ability to the AWS access portal 13
  - Set up single sign-on access for the SDK 13
  - Sign in using the AWS CLI 13
  - Install Java and a build tool 14
  - Additional authentication options 14
- Build tools setup 14
  - Set up an Apache Maven project 14
  - Set up a Gradle project 18
- Set up a GraalVM Native Image project 19
  - Prerequisites 19
  - Create a project using the archetype 19
  - Build a native image 20
- Migrate to version 2 20
  - What's new 20
  - What's different between 1.x and 2.x 21
  - Use the SDK for Java 1.x and 2.x side-by-side 30

Use the SDK 32
- Creating service clients 32
  - Using the default client 32
  - Making requests 32
  - Handling responses 33
  - Closing the client 33
- Handling exceptions 33
- Using waiters 34
- Configuring service clients 34
- HTTP clients 35
- Retries 35
- Timeouts 35
- Execution interceptors 35
- Additional information 35
- Provide temporary credentials to the SDK 35
  - Use temporary credentials 36
  - Default credentials provider chain 37
  - Use a specific credentials provider or provider chain 38
  - Use profiles 38
  - Load temporary credentials from an external process 40

Supply temporary credentials in code ... 43
Configuring IAM roles for Amazon EC2 ... 44
AWS region selection ... 45
Choosing a region ... 45
Choosing a specific endpoint ... 46
Automatically determine the Region from the environment ... 46
Checking for service availability in a Region ... 47
Reducing SDK startup time for AWS Lambda ... 47
Use the SDK's `UrlConnectionHttpClient` ... 47
Use the SDK's `AwsCrtAsyncHttpClient` ... 47
Remove unused HTTP client dependencies ... 48
Configure service clients to shortcut lookups ... 49
Initialize the SDK client outside of the Lambda function handler ... 49
Minimize dependency injection ... 50
Use a Maven Archetype targeting AWS Lambda ... 50
Lambda SnapStart ... 50
Version 2.x changes that affect startup time ... 50
Additional resources ... 50
Configuring HTTP clients ... 50
HTTP clients in the SDK ... 50
Smart configuration defaults ... 51
Configuring the Apache-based HTTP client ... 52
Configuring the URLConnection-based HTTP client ... 55
Configuring the Netty-based HTTP client ... 59
Configuring the AWS CRT-based HTTP client ... 62
Exception handling ... 66
Why unchecked exceptions? ... 67
AwsServiceException (and subclasses) ... 67
SdkClientException ... 67
Logging ... 68
Log4j 2 configuration file ... 68
Setting dependencies ... 68
SDK-specific errors and warnings ... 69
Request/response summary logging ... 69
Verbose wire logging ... 70
Setting the JVM TTL for DNS name lookups ... 73
How to set the JVM TTL ... 73
Best practices ... 74
Reuse an SDK client ... 74
Close input streams ... 74
Tune HTTP configurations ... 74
Use OpenSSL for Netty ... 75
Configure API timeouts ... 75
Use metrics ... 76
SDK features ... 77
Asynchronous programming ... 77
Non-streaming operations ... 77
Streaming operations ... 79
Advanced operations ... 82
DynamoDB Enhanced Client ... 83
HTTP/2 ... 83
SDK Metrics ... 83
Prerequisites ... 83
How to enable metrics collection ... 84
What information is collected? ... 85
How can I use this information? ... 85
Service client metrics ... 85

Pagination ... 88
Synchronous pagination ... 88
Asynchronous pagination ... 90
AWS CRT-based S3 client ... 94
Add dependencies ... 94
Create an instance ... 95
Use the AWS CRT-based S3 client ... 95
S3 Transfer Manager ... 95
Get started ... 96
Upload a file ... 97
Download a file ... 98
Copy an object ... 98
Upload a directory ... 99
Download to a directory ... 100
Waiters ... 101
Prerequisites ... 101
Using waiters ... 101
Configuring waiters ... 102
Code examples ... 103
Work with AWS services ... 104
Guided code examples ... 104
AWS database services ... 105
Amazon Simple Storage Service (S3) ... 106
Amazon DynamoDB ... 120
Amazon EC2 ... 141
AWS Identity and Access Management (IAM) ... 160
Amazon Athena ... 177
Amazon CloudWatch ... 177
AWS CloudTrail ... 187
Amazon Cognito ... 187
Amazon Kinesis Data Firehose ... 192
Amazon Forecast ... 192
Amazon Kinesis ... 192
AWS Lambda ... 198
AWS Elemental MediaConvert ... 201
AWS Elemental MediaStore ... 201
AWS Migration Hub ... 201
Amazon Pinpoint ... 201
Amazon Polly ... 208
Amazon SageMaker ... 209
Amazon Simple Notification Service ... 209
Amazon Simple Queue Service (SQS) ... 213
AWS Systems Manager ... 217
Amazon Simple Workflow Service ... 218
Amazon Transcribe ... 218
Amazon Translate ... 222
Amazon WorkDocs ... 222
Code examples ... 222
Actions and scenarios ... 222
Cross-service examples ... 656
Security ... 661
Data protection ... 661
Transport Layer Security (TLS) ... 662
Check TLS versions ... 662
Enforce TLS versions ... 663
Migrate to TLS 1.2 ... 663
Identity and Access Management ... 663

Audience 663
Authenticating with identities 663
Compliance Validation 666
Resilience 666
Infrastructure Security 667
Document history 668

Developer Guide - AWS SDK for Java 2.x

The AWS SDK for Java provides a Java API for AWS services. Using the SDK, you can build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

The AWS SDK for Java 2.x is a major rewrite of the version 1.x code base. It's built on top of Java 8+ and adds several frequently requested features. These include support for non-blocking I/O and the ability to plug in a different HTTP implementation at runtime.

We regularly add support for new services to the AWS SDK for Java. For a list of changes and features in a particular version, view the change log.

Get started with the SDK

If you're ready to get hands-on with the SDK, follow the *Get started tutorial* (p. 3) tutorial.

To set up your development environment, see *Setup* (p. 11).

If you're currently using version 1.x of the SDK for Java, see Migrate to version 2 (p. 20) for specific guidance.

For information on making requests to Amazon S3, DynamoDB, Amazon EC2 and other AWS services, see Use the SDK for Java (p. 32) and Work with AWS services (p. 104).

Develop mobile applications

If you're a mobile app developer, Amazon Web Services provides the AWS Amplify framework. For more information, see the Amplify Documentation.

Maintenance and support for SDK major versions

For information about maintenance and support for SDK major versions and their underlying dependencies, see the following topics in the AWS SDKs and Tools Reference Guide:

- AWS SDKs and Tools Maintenance Policy
- AWS SDKs and Tools Version Support Matrix

Additional resources

In addition to this guide, the following are valuable online resources for AWS SDK for Java developers:

- AWS SDK for Java 2.x API Reference
- Java developer blog

- Java development topic in AWS re:Post
- GitHub:
  - Developer Guide documentation source
  - SDK source
- The AWS Code Sample Catalog
- @awsforjava (Twitter)

Features not yet in version 2 of the SDK

For details about additional features that aren't yet in 2.x, see the following GitHub issues. Comments and feedback are also welcome.

- High-level libraries
  - Amazon S3 Encryption Client
  - DynamoDB document APIs
  - DynamoDB Encryption Client
  - Amazon SQS Client-side Buffering

Contributing to the SDK

Developers can also contribute feedback through the following channels:

- Submit issues on GitHub:
  - Submit Developer Guide documentation issues
  - Submit SDK issues
- Join an informal chat about the SDK on the AWS SDK for Java 2.x gitter channel

Get started with the AWS SDK for Java 2.x

The AWS SDK for Java 2.x provides Java APIs for Amazon Web Services (AWS). Using the SDK, you can build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

This tutorial shows you how to use Apache Maven to define dependencies for the SDK for Java 2.x and then write code that connects to Amazon S3 to upload a file.

Follow these steps to complete this tutorial:

- Step 1: Set up for this tutorial (p. 3)
- Step 2: Create the project (p. 3)
- Step 3: Write the code (p. 6)
- Step 4: Build and run the application (p. 9)

Step 1: Set up for this tutorial

Before you begin this tutorial, you need the following:

- Permission to access Amazon S3
- A Java development environment that is configured to access AWS services using single sign-on to the AWS IAM Identity Center (successor to AWS Single Sign-On)

Use the instructions in Basic setup (p. 12) to get set up for this tutorial. After you have configured your development environment with single sign-on access (p. 13) for the Java SDK and you have an active AWS access portal session (p. 13), continue with Step 2 of this tutorial.

Step 2: Create the project

To create the project for this tutorial, you run a Maven command that prompts you for input on how to configure the project. After all input is entered and confirmed, Maven finishes building out the project by creating a `pom.xml` and creates stub Java files.

1. Open a terminal or command prompt window and navigate to a directory of your choice, for example, your `Desktop` or `Home` folder.
2. Enter the following command at the terminal and press `Enter`.

```
mvn archetype:generate \
  -DarchetypeGroupId=software.amazon.awssdk \
  -DarchetypeArtifactId=archetype-app-quickstart \
  -DarchetypeVersion=2.18.16
```

3. Enter the value listed in the second column for each prompt.

Prompt	Value to enter
`Define value for property 'service':`	`s3`
`Define value for property 'httpClient':`	`apache-client`
`Define value for property 'nativeImage':`	`false`
`Define value for property 'groupId':`	`org.example`
`Define value for property 'artifactId':`	`getstarted`
`Define value for property 'version' 1.0-SNAPSHOT:`	*<Enter>*
`Define value for property 'package' org.example:`	*<Enter>*

4. After the last value is entered, Maven lists the choices you made. Confirm by entering *Y* or re-enter values by entering *N*.

Maven creates the project folder named `getstarted` based on the `artifactId` value that you entered. Inside the `getstarted` folder, find a `README.md` file that you can review, a `pom.xml` file, and a `src` directory.

Maven builds the following directory tree.

```
getstarted
### README.md
### pom.xml
### src
    ### main
    #   ### java
    #   #   ### org
    #   #       ### example
    #   #           ### App.java
    #   #           ### DependencyFactory.java
    #   #           ### Handler.java
    #   ### resources
    #       ### simplelogger.properties
    ### test
        ### java
            ### org
                ### example
                    ### HandlerTest.java

10 directories, 7 files
```

The following shows the contents of the `pom.xml` project file.

`pom.xml`

The `dependencyManagement` section contains a dependency to the AWS SDK for Java 2.x and the `dependencies` section has a dependency for Amazon S3. The project uses Java 1.8 because of the `1.8` value in the `maven.compiler.source` and `maven.compiler.target` properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>getstarted</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.shade.plugin.version>3.2.1</maven.shade.plugin.version>
        <maven.compiler.plugin.version>3.6.1</maven.compiler.plugin.version>
        <exec-maven-plugin.version>1.6.0</exec-maven-plugin.version>
        <aws.java.sdk.version>2.18.16</aws.java.sdk.version>
        <slf4j.version>1.7.28</slf4j.version>
        <junit5.version>5.8.1</junit5.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.java.sdk.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>netty-nio-client</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>apache-client</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>commons-logging</groupId>
                    <artifactId>commons-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
```

```
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>${slf4j.version}</version>
        </dependency>

        <!-- Needed to adapt Apache Commons Logging used by Apache HTTP Client to Slf4j to
 avoid
        ClassNotFoundException: org.apache.commons.logging.impl.LogFactoryImpl during
 runtime -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <version>${slf4j.version}</version>
        </dependency>

        <!-- Test Dependencies -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>${junit5.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>${maven.compiler.plugin.version}</version>
            </plugin>
        </plugins>
    </build>

</project>
```

Copy the following two dependencies and paste them inside the `<dependencies>` section of the `pom.xml` file.

```
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sso</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>ssooidc</artifactId>
        </dependency>
```

Step 3: Write the code

The following code shows the `App` class created by Maven. The `main` method is the entry point into the application, which creates an instance of the `Handler` class and then calls its `sendRequest` method.

App class

```
package org.example;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class App {
    private static final Logger logger = LoggerFactory.getLogger(App.class);

    public static void main(String... args) {
        logger.info("Application starts");

        Handler handler = new Handler();
        handler.sendRequest();

        logger.info("Application ends");
    }
}
```

The `DependencyFactory` class created by Maven contains the `s3Client` factory method that build and returns an `S3Client` instance. The `S3Client` instance uses an instance of the Apache-based HTTP client. This is because you specified `apache-client` when Maven prompted you for which HTTP client to use.

The `DependencyFactory` is shown in the following code.

`DependencyFactory` class

```
package org.example;

import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;

/**
 * The module containing all dependencies required by the {@link Handler}.
 */
public class DependencyFactory {

    private DependencyFactory() {}

    /**
     * @return an instance of S3Client
     */
    public static S3Client s3Client() {
        return S3Client.builder()
                       .httpClientBuilder(ApacheHttpClient.builder())
                       .build();
    }
}
```

The `Handler` class contains the main logic of your program. When an instance of `Handler` is created in the App class, the `DependencyFactory` furnishes the `S3Client` service client. Your code uses the `S3Client` instance to call the Amazon S3 service.

Maven generates the following `Handler` class with a *TODO* comment. The next step in the tutorial replaces the *TODO* with code.

`Handler` class, Maven-generated

```
package org.example;

import software.amazon.awssdk.services.s3.S3Client;
```

```
public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        // TODO: invoking the api calls using s3Client.
    }
}
```

To fill in the logic, replace the entire contents of the `Handler` class with the following code. The `sendRequest` method is filled in and the necessary imports are added.

`Handler` class, implemented

The code first creates a new S3 bucket with the last part of the name generated using `System.currentTimeMillis()` in order to make the bucket name unique.

After creating the bucket in the `createBucket()` method, the program uploads an object using the `putObject` method of `S3Client`. The contents of the object is a simple string created with the `RequestBody.fromString` method.

Finally, the program deletes the object followed by the bucket in the `cleanUp` method.

```
package org.example;

import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;


public class Handler {
    private final S3Client s3Client;

    public Handler() {
        s3Client = DependencyFactory.s3Client();
    }

    public void sendRequest() {
        String bucket = "bucket" + System.currentTimeMillis();
        String key = "key";

        createBucket(s3Client, bucket);

        System.out.println("Uploading object...");

        s3Client.putObject(PutObjectRequest.builder().bucket(bucket).key(key)
                        .build(),
                RequestBody.fromString("Testing with the {sdk-java}"));

        System.out.println("Upload complete");
        System.out.printf("%n");

        cleanUp(s3Client, bucket, key);
```

```
        System.out.println("Closing the connection to {S3}");
        s3Client.close();
        System.out.println("Connection closed");
        System.out.println("Exiting...");
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            s3Client.createBucket(CreateBucketRequest
                    .builder()
                    .bucket(bucketName)
                    .build());
            System.out.println("Creating bucket: " + bucketName);
            s3Client.waiter().waitUntilBucketExists(HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build());
            System.out.println(bucketName + " is ready.");
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void cleanUp(S3Client s3Client, String bucketName, String keyName) {
        System.out.println("Cleaning up...");
        try {
            System.out.println("Deleting object: " + keyName);
            DeleteObjectRequest deleteObjectRequest =
 DeleteObjectRequest.builder().bucket(bucketName).key(keyName).build();
            s3Client.deleteObject(deleteObjectRequest);
            System.out.println(keyName + " has been deleted.");
            System.out.println("Deleting bucket: " + bucketName);
            DeleteBucketRequest deleteBucketRequest =
 DeleteBucketRequest.builder().bucket(bucketName).build();
            s3Client.deleteBucket(deleteBucketRequest);
            System.out.println(bucketName + " has been deleted.");
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Cleanup complete");
        System.out.printf("%n");
    }
}
```

Step 4: Build and run the application

After the project is created and contains the complete `Handler` class, build and run the application.

1. Make sure that you have an active IAM Identity Center session. To do so, run the AWS Command Line Interface command `aws sts get-caller-identity` and check the response. If you don't have an active session, see this section (p. 13) for instructions.
2. Open a terminal or command prompt window and navigate to your project directory `getstarted`.
3. Use the following command to build your project:

```
mvn clean package
```

4. Use the following command to run the application.

```
mvn exec:java -Dexec.mainClass="org.example.App"
```

To view the new bucket and object that the program creates, perform the following steps.

1. In `Handler.java`, comment out the line `cleanUp(s3Client, bucket, key)` in the `sendRequest` method and save the file.
2. Rebuild the project by running `mvn clean package`.
3. Rerun `mvn exec:java -Dexec.mainClass="org.example.App"` to upload the text object once more.
4. Sign in to the S3 console to view the new object in the newly created bucket.

After you view the file, delete the object, and then delete the bucket.

Success

If your Maven project built and ran without error, then congratulations! You have successfully built your first Java application using the SDK for Java 2.x.

Cleanup

To clean up the resources you created during this tutorial, do the following:

- If you haven't done so already, in the S3 console, delete any objects and any buckets created when you ran the application.
- In the IAM console, delete the *TestSDK* user.

  If you delete this user, also remove the contents of the `credentials` file you created during setup.
- Delete the project folder (`getstarted`).

Next steps

Now that you have the basics down, you can learn about the following:

- Working with Amazon S3 (p. 106)
- Working with other Amazon Web Services (p. 104), such as DynamoDB (p. 120), Amazon EC2 (p. 141), and various database services (p. 105)
- Use the SDK (p. 32)
- Security for the AWS SDK for Java (p. 661)

Set up the AWS SDK for Java 2.x

The AWS SDK for Java 2.x provides Java APIs for Amazon Web Services (AWS). Using the SDK, you can build Java applications that work with Amazon S3, Amazon EC2, DynamoDB, and more.

This section provides information about how to set up your development environment and projects to use the AWS SDK for Java 2.x.

Contents of this chapter

- Basic setup to work with AWS services (p. 12)
  - Overview (p. 12)
  - Sign-in ability to the AWS access portal (p. 13)
  - Set up single sign-on access for the SDK (p. 13)
  - Sign in using the AWS CLI (p. 13)
  - Install Java and a build tool (p. 14)
  - Additional authentication options (p. 14)
- Build tools setup (p. 14)
  - Set up an Apache Maven project (p. 14)
    - Prerequisites (p. 14)
    - Create a Maven project (p. 15)
    - Configure the Java compiler for Maven (p. 15)
    - Declare the SDK as a dependency (p. 16)
    - Set dependencies for SDK modules (p. 16)
      - Build the entire SDK into your project (p. 17)
    - Build your project (p. 18)
  - Set up a Gradle project (p. 18)
- Set up a GraalVM Native Image project for the AWS SDK for Java (p. 19)
  - Prerequisites (p. 19)
  - Create a project using the archetype (p. 19)
  - Build a native image (p. 20)
- Migrate from version 1.x to 2.x of the AWS SDK for Java (p. 20)
  - What's new (p. 20)
  - What's different between the AWS SDK for Java 1.x and 2.x (p. 21)
    - Package name change (p. 21)
    - High-level libraries (p. 21)
    - Adding version 2.x to your project (p. 22)
    - Client builders (p. 22)
    - Client Configuration (p. 22)
    - Setter Methods (p. 23)
    - Class Names (p. 23)
    - Region class (p. 24)
    - Immutable POJOs (p. 24)
    - Streaming operations (p. 24)
    - Exception changes (p. 25)
    - Service-specific changes (p. 25)

- Amazon S3 operation name changes (p. 25)
- Cross-Region access (p. 26)
- Additional client changes (p. 26)
  - Default client changes (p. 26)
- Credentials provider changes (p. 27)
  - Credentials provider (p. 27)
  - Credentials provider changes mapped between versions 1.x and 2.x (p. 27)
- Region class name changes (p. 28)
  - Region configuration (p. 28)
  - Method and class name mappings (p. 28)
- Exception class name changes (p. 29)
- Use the SDK for Java 1.x and 2.x side-by-side (p. 30)

Basic setup to work with AWS services

Overview

To successfully develop applications that access AWS services using the AWS SDK for Java, the following conditions are required:

- You must be able to sign in to the AWS access portal (p. 13) available in the AWS IAM Identity Center (successor to AWS Single Sign-On).
- The permissions of the IAM role configured for the SDK must allow access to the AWS services that your application requires. The permissions associated with the **PowerUserAccess** AWS managed policy are sufficient for most development needs.
- A development environment with the following elements:
  - Shared configuration files that are set up in at least one of the following ways:
    - The `config` file contains IAM Identity Center single sign-on settings (p. 13) so that the SDK can get AWS credentials.
    - The `credentials` file contains temporary credentials.
  - An installation of Java 8 (p. 14) or later.
  - A build automation tool (p. 14)such as Maven or Gradle.
  - A text editor to work with code.
  - (Optional, but recommended) An IDE (integrated development environment) such as IntelliJ IDEA, Eclipse, or NetBeans.

    When you use an IDE, you can also integrate AWS Toolkits to more easily work with AWS services. The AWS Toolkit for IntelliJ and AWS Toolkit for Eclipse are two toolkits that you can use for Java development.

- An active AWS access portal session when you are ready to run your application. You use the AWS Command Line Interface to initiate the sign-in process (p. 13) to IAM Identity Center's AWS access portal.

  Important
  The instructions in this setup section assume that you or organization uses IAM Identity Center. If your organization uses an external identity provider that works independently of IAM Identity

Center, find out how you can get temporary credentials for the SDK for Java to use. Follow these instructions (p. 36) to add temporary credentials to the `~/.aws/credentials` file.
If your identity provider adds temporary credentials automatically to the `~/.aws/credentials` file, make sure that the profile name is `[default]` so that you do not need to provide a profile name to the SDK or AWS CLI.

Sign-in ability to the AWS access portal

The AWS access portal is the web location where you manually sign in to the IAM Identity Center. The format of the URL is `d-xxxxxxxxxx.awsapps.com/start`or *`your_subdomain`*`.awsapps.com/start`. If you are not familiar with the AWS access portal, follow the guidance for account access in the IAM Identity Center authentication topic in the AWS SDKs and Tools Reference Guide.

Set up single sign-on access for the SDK

After you complete Step 2 in the programmatic access section in order for the SDK to use IAM Identity Center authentication, your system should contain the following elements.

- The AWS CLI, which you use to start an AWS access portal session (p. 13) before you run your application.
- An `~/.aws/config` file that contains a default profile. The SDK for Java uses the profile's SSO token provider configuration to acquire credentials before sending requests to AWS. The `sso_role_name` value, which is an IAM role connected to an IAM Identity Center permission set, should allow access to the AWS services used in your application.

  The following sample `config` file shows a default profile set up with SSO token provider configuration. The profile's `sso_session` setting refers to the named `sso-session` section. The `sso-session` section contains settings to initiate an AWS access portal session.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

For more details about the settings used in the SSO token provider configuration, see SSO token provider configuration in the AWS SDKs and Tools Reference Guide.

If your development environment is not set up for programmatic access as previously shown, follow Step 2 in the SDKs Reference Guide.

Sign in using the AWS CLI

Before running an application that accesses AWS services, you need an active AWS access portal session in order for the SDK to use IAM Identity Center authentication to resolve credentials. Run the following command in the AWS CLI to sign in to the AWS access portal.

```
aws sso login
```

Since you have a default profile setup, you do not need to call the command with a `--profile` option. If your SSO token provider configuration is using a named profile, the command is `aws sso login --profile named-profile`.

To test if you already have an active session, run the following AWS CLI command.

```
aws sts get-caller-identity
```

The response to this command should report the IAM Identity Center account and permission set configured in the shared `config` file.

> **Note**
> If you already have an active AWS access portal session and run `aws sso login`, you will not be required to provide credentials.
> However, you will see a dialog that requests permission for `botocore` to access your information. `botocore` is the foundation for the AWS CLI .
> Select **Allow** to authorize access to your information for the AWS CLI and SDK for Java.

Install Java and a build tool

Your development environment needs the following:

- Java 8 or later. The AWS SDK for Java works with the Oracle Java SE Development Kit and with distributions of Open Java Development Kit (OpenJDK) such as Amazon Corretto, Red Hat OpenJDK, and AdoptOpenJDK.
- A build tool or IDE that supports Maven Central such as Apache Maven, Gradle, or IntelliJ.
  - For information about how to install and use Maven, see http://maven.apache.org/.
  - For information about how to install and use Gradle, see https://gradle.org/.
  - For information about how to install and use IntelliJ IDEA, see https://www.jetbrains.com/idea/.

Additional authentication options

For more options on authentication for the SDK, such as the use of profiles and environment variables, see the configuration chapter in the AWS SDKs and Tools Reference Guide.

Build tools setup

Set up an Apache Maven project

You can use Apache Maven to set up and build AWS SDK for Java projects, or to build the SDK itself.

Prerequisites

To use the AWS SDK for Java with Maven, you need the following:

- Java *8.0 or later*. You can download the latest Java SE Development Kit software from http://www.oracle.com/technetwork/java/javase/downloads/. The AWS SDK for Java also works with OpenJDK and Amazon Corretto, a distribution of the Open Java Development Kit (OpenJDK). Download the latest OpenJDK version from https://openjdk.java.net/install/index.html. Download the latest Amazon Corretto 8 or Amazon Corretto 11 version from the Corretto page.
- *Apache Maven*. If you need to install Maven, go to http://maven.apache.org/ to download and install it.

Create a Maven project

To create a Maven project from the command line, run the following command from a terminal or command prompt window.

```
mvn -B archetype:generate \
 -DarchetypeGroupId=software.amazon.awssdk \
 -DarchetypeArtifactId=archetype-lambda -Dservice=s3 -Dregion=US_WEST_2 \
 -DgroupId=com.example.myapp \
 -DartifactId=myapp
```

> **Note**
> Replace *com.example.myapp* with the full package namespace of your application. Also replace *myapp* with your project name. This becomes the name of the directory for your project.

This command creates a Maven project using the archetype templating toolkit. The archetype generates the scaffolding for an AWS Lambda function handler project . This project archetype is preconfigured to compile with Java SE 8 and includes a dependency to the SDK for Java 2.x.

For more information about creating and configuring Maven projects, see the Maven Getting Started Guide.

Configure the Java compiler for Maven

If you created your project using the AWS Lambda project archetype as described earlier, this is already done for you.

To verify that this configuration is present, start by opening the `pom.xml` file from the project folder you created (for example, `myapp`) when you executed the previous command. Look on lines 11 and 12 to see the Java compiler version setting for this Maven project, and the required inclusion of the Maven compiler plugin on lines 71-75.

```
<project>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven.compiler.plugin.version}</version>
      </plugin>
    </plugins>
  </build>
</project>
```

If you create your project with a different archetype or by using another method, you must ensure that the Maven compiler plugin is part of the build and that its source and target properties are both set to **1.8** in the `pom.xml` file.

See the previous snippet for one way to configure these required settings.

Alternatively, you can configure the compiler configuration inline with the plugin declaration, as follows.

```
<project>
  <build>
```

```
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Declare the SDK as a dependency

To use the AWS SDK for Java in your project, you need to declare it as a dependency in your project's `pom.xml` file.

If you created your project using the project archetype as described earlier, the SDK is already configured as a dependency in your project. We recommend that you update this configuration to reference the latest version of the AWS SDK for Java. To do so, open the `pom.xml` file and change the `aws.java.sdk.version` property (on line 16) to the latest version. The following is an example.

```
<project>
  <properties>
    <aws.java.sdk.version>2.20.20</aws.java.sdk.version>
  </properties>
</project>
```

Find the latest version of the AWS SDK for Java in the AWS SDK for Java API Reference version 2.x. The version is listed in the title of the page.

If you created your Maven project in a different way, configure the latest version of the SDK for your project by ensuring that the `pom.xml` file contains the following.

```
<project>
  <properties>
    <aws.java.sdk.version>2.20.20</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

Note
Replace *2.X.X* in the `pom.xml` file with a valid version of the AWS SDK for Java version 2.

Set dependencies for SDK modules

Now that you have configured the SDK, you can add dependencies for one or more of the AWS SDK for Java modules to use in your project.

Although you can specify the version number for each component, you don't need to because you already declared the SDK version in the `dependencyManagement` section. To load a custom version of a given module, specify a version number for its dependency.

If you created your project using the project archetype as described earlier, your project is already configured with multiple dependencies. These include dependences for Lambda function handlers and Amazon S3, as follows.

```
<project>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
    </dependency>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-lambda-java-core</artifactId>
      <version>1.2.0</version>
    </dependency>
  </dependencies>
</project>
```

Note

In the `pom.xml` example above, the dependencies are from different `groupIds`. The `s3` dependency is from `software.amazon.awssdk`, whereas the `aws-lambda-java-core` dependency is from `com.amazonaws`. The `bom` dependency management configuration affects artifacts for `software.amazon.awssdk`, so a version is needed for the `aws-lambda-java-core` artifact.

For the development of *Lambda function handlers* using the SDK for Java 2.x, `aws-lambda-java-core` is the correct dependency. However, if your application needs to manage Lambda resources, using operations such as `listFunctions`, `deleteFunction`, `invokeFunction`, and `createFunction`, your application requires the following dependency.

```
<groupId>software.amazon.awssdk</groupId>
<artifactId>lambda</artifactId>
```

Add the modules to your project for the AWS service and features you need for your project. The modules (dependencies) that are managed by the AWS SDK for Java BOM are listed on the Maven central repository (https://mvnrepository.com/artifact/software.amazon.awssdk/bom/latest).

Note

You can look at the `pom.xml` file from a code example to determine which dependencies you need for your project. For example, if you're interested in the dependencies for the Amazon S3 service, see this example from the AWS Code Examples Repository on GitHub. (Look for the `pom.xml` file under /java2/example_code/s3.)

Build the entire SDK into your project

To optimize your application, we strongly recommend that you pull in only the components you need instead of the entire SDK. However, to build the entire AWS SDK for Java into your project, declare it in your `pom.xml` file, as follows.

```
<project>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-sdk-java</artifactId>
      <version>2.X.X</version>
```

```
    </dependency>
  </dependencies>
</project>
```

Build your project

After you configure the `pom.xml` file, you can use Maven to build your project.

To build your Maven project from the command line, open a terminal or command prompt window, navigate to your project directory (for example, `myapp`), enter or paste the following command, then press Enter or Return.

```
mvn package
```

This creates a single `.jar` file (JAR) in the `target` directory (for example, `myapp/target`). This JAR contains all of the SDK modules you specified as dependencies in your `pom.xml` file.

Set up a Gradle project

You can use Gradle to set up and build AWS SDK for Java projects.

To manage SDK dependencies for your Gradle project, import the Maven bill of materials (BOM) for the AWS SDK for Java into the `build.gradle` file.

> **Note**
> In the following examples, replace *2.15.0* in the `build.gradle` file with the latest version of the AWS SDK for Java v2. Find the latest version in the AWS SDK for Java API Reference version 2.x.

1. Add the BOM to the *dependencies* section of the file.

```
...
dependencies {
 implementation platform('software.amazon.awssdk:bom:2.15.0')

 // Declare individual SDK dependencies without version.
 ...
}
```

2. Specify the SDK modules to use in the *dependencies* section. For example, the following includes a dependency for Amazon Kinesis.

```
...
dependencies {
 ...
 implementation 'software.amazon.awssdk:kinesis'
 ...
}
```

Gradle automatically resolves the correct version of your SDK dependencies by using the information from the BOM.

The following is an example of a complete `build.gradle` file that includes a dependency for Kinesis.

```
group 'aws.test'
version '1.0'
```

```
apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
 mavenCentral()
}

dependencies {
 implementation platform('software.amazon.awssdk:bom:2.15.0')
 implementation 'software.amazon.awssdk:kinesis'
 testImplementation group: 'junit', name: 'junit', version: '4.11'
}
```

Note
In the previous example, replace the dependency for Kinesis with the dependencies of the AWS services you will use in your project. The modules (dependencies) that are managed by the AWS SDK for Java BOM are listed on the Maven central repository.

For more information about specifying SDK dependencies by using the BOM, see Setting up an Apache Maven project (p. 14).

Set up a GraalVM Native Image project for the AWS SDK for Java

With versions 2.16.1 and later, the AWS SDK for Java provides out-of-the-box support for GraalVM Native Image applications. Use the `archetype-app-quickstart` Maven archetype to set up a project with built-in native image support.

Prerequisites

- Complete the steps in Setting up the AWS SDK for Java 2.x (p. 11).
- Install GraalVM Native Image.

Create a project using the archetype

To create a Maven project with built-in native image support, in a terminal or command prompt window, use the following command.

Note
Replace `com.example.mynativeimageapp` with the full package namespace of your application. Also replace `mynativeimageapp` with your project name. This becomes the name of the directory for your project.

```
mvn archetype:generate \
    -DarchetypeGroupId=software.amazon.awssdk \
    -DarchetypeArtifactId=archetype-app-quickstart \
    -DarchetypeVersion=2.16.1 \
    -DnativeImage=true \
    -DhttpClient=apache-client \
    -Dservice=s3 \
    -DgroupId=com.example.mynativeimageapp \
    -DartifactId=mynativeimageapp \
```

```
    -DinteractiveMode=false
```

This command creates a Maven project configured with dependencies for the AWS SDK for Java, Amazon S3, and the `ApacheHttpClient` HTTP client. It also includes a dependency for the GraalVM Native Image Maven plugin, so that you can build native images using Maven.

To include dependencies for a different Amazon Web Services, set the value of the `-Dservice` parameter to the artifact ID of that service. Examples include `dynamodb`, `comprehend`, and `pinpoint`. For a complete list of artifact IDs, see the list of managed dependencies for software.amazon.awssdk on Maven Central.

To use an asynchronous HTTP client, set the `-DhttpClient` parameter to `netty-nio-client`. To use `UrlConnectionHttpClient` as the synchronous HTTP client instead of `apache-client`, set the `-DhttpClient` parameter to `url-connection-client`.

Build a native image

After you create the project, run the following command from your project directory, for example, `mynativeimageapp`:

```
mvn package -P native-image
```

This creates a native image application in the `target` directory, for example, `target/mynativeimageapp`.

Migrate from version 1.x to 2.x of the AWS SDK for Java

The AWS SDK for Java 2.x is a major rewrite of the 1.x code base built on top of Java 8+. It includes many updates, such as improved consistency, ease of use, and strongly enforced immutability. This section describes the major features that are new in version 2.x, and provides guidance on how to migrate your code to version 2.x from 1.x.

Topics

- What's new (p. 20)
- What's different between the AWS SDK for Java 1.x and 2.x (p. 21)
- Use the SDK for Java 1.x and 2.x side-by-side (p. 30)

What's new

- You can configure your own HTTP clients. See HTTP transport configuration (p. 50).
- Async clients are now truly nonblocking and return `CompletableFuture` objects. See Asynchronous programming (p. 77).
- Operations that return multiple pages have autopaginated responses. This way, you can focus your code on what to do with the response, without the need to check for and get subsequent pages. See Pagination (p. 88).
- SDK start time performance for AWS Lambda functions is improved. See SDK start time performance improvements (p. 47).
- Version 2.x supports a new shorthand method for creating requests.

Example

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```

For more details about the new features and to see specific code examples, refer to the other sections of this guide.

- Quick Start (p. 3)
- Setting up (p. 11)
- Code examples for the AWS SDK for Java 2.x (p. 104)
- Use the SDK (p. 32)
- Security for the AWS SDK for Java (p. 661)

What's different between the AWS SDK for Java 1.x and 2.x

This section describes the main changes to be aware of when converting an application from using the AWS SDK for Java version 1.x to version 2.x.

Package name change

A noticable change from the the SDK for Java 1.x to the SDK for Java 2.x is the package name change. Package names begin with `software.amazon.awssdk` in SDK 2.x, whereas the SDK 1.x uses `com.amazonaws`.

These same names differentiate Maven artifacts from SDK 1.x to SDK 2.x. Maven artifacts for the SDK 2.x use the `software.amazon.awssdk` groupId, whereas the SDK 1.x uses the `com.amazonaws` groupId.

There are a few times when your code requires a `com.amazonaws` dependency for a project that otherwise uses only SDK 2.x artifacts. One example of this is when you work with server-side AWS Lambda. This was shown in the Set up an Apache Maven project (p. 16) section earlier in this guide.

> **Note**
> Several package names in the SDK 1.x contain `v2`. The use of `v2` in this case usually means that code in the package is targted to work with version 2 of the service.
> Since the full package name begins with `com.amazonaws`, these are SDK 1.x components. Examples of these package names in the SDK 1.x are:
>
> - `com.amazonaws.services.dynamodbv2`
> - `com.amazonaws.retry.v2`
> - `com.amazonaws.services.apigatewayv2`
> - `com.amazonaws.services.simpleemailv2`

High-level libraries

High-level libraries, such as Amazon SQS Client-side Buffering, are not yet available in version 2.x. For a list of libraries that have been released, see the AWS SDK for Java 2.x changelog.

If your application depends on libraries that aren't released in version 2.x, configure your `pom.xml` file to use both 1.x and 2.x. For more information, see Using both SDKs side-by-side (p. 30).

Adding version 2.x to your project

Maven is the recommended way to manage dependencies when using the AWS SDK for Java 2.x. To add version 2 components to your project, update your `pom.xml` file with a dependency on the SDK.

Example

```
<dependencyManagement>
    <dependencies>
        <dependency>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>bom</artifactId>
          <version>2.16.1</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```

Client builders

You must create all clients using the client builder method. Constructors are no longer available.

Example of creating a client in version 1.x

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

Example of creating a client in version 2.x

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

Client Configuration

In 1.x, SDK client configuration was modified by setting a `ClientConfiguration` instance on the client or client builder. In version 2.x, the client configuration is split into separate configuration classes. With the separate configuration classes, you can configure different HTTP clients for async versus synchronous clients but still use the same `ClientOverrideConfiguration` class.

Example of client configuration in version 1.x

```
AmazonDynamoDBClientBuilder.standard()
.withClientConfiguration(clientConfiguration)
.build()
```

Example of synchronous client configuration in version 2.x

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();
```

```
ApacheHttpClient.Builder httpClientBuilder =
        ApacheHttpClient.builder()
                        .proxyConfiguration(proxyConfig.build());

ClientOverrideConfiguration.Builder overrideConfig =
        ClientOverrideConfiguration.builder();

DynamoDbClient client =
        DynamoDbClient.builder()
                      .httpClientBuilder(httpClientBuilder)
                      .overrideConfiguration(overrideConfig.build())
                      .build();
```

Example of asynchronous client configuration in version 2.x

```
NettyNioAsyncHttpClient.Builder httpClientBuilder =
        NettyNioAsyncHttpClient.builder();

ClientOverrideConfiguration.Builder overrideConfig =
        ClientOverrideConfiguration.builder();

ClientAsyncConfiguration.Builder asyncConfig =
        ClientAsyncConfiguration.builder();

DynamoDbAsyncClient client =
        DynamoDbAsyncClient.builder()
                           .httpClientBuilder(httpClientBuilder)
                           .overrideConfiguration(overrideConfig.build())
                           .asyncConfiguration(asyncConfig.build())
                           .build();
```

For a complete mapping of client configuration methods between 1.x and 2.x, see the AWS SDK for Java 2.x changelog.

Setter Methods

In the AWS SDK for Java 2.x, setter method names don't include the `set` or `with` prefix. For example, `*.withEndpoint()` is now `*.endpoint()`.

Example of using setting methods in 1.x

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
          .withRegion("us-east-1")
          .build();
```

Example of using setting methods in 2.x

```
DynamoDbClient client = DynamoDbClient.builder()
          .region(Region.US_EAST_1)
          .build();
```

Class Names

All client class names are now fully camel cased and no longer prefixed by `Amazon`. These changes are aligned with names used in the AWS CLI. For a full list of client name changes, see the AWS SDK for Java 2.x changelog.

Example of class names in 1.x

```
AmazonDynamoDB
AWSACMPCAAsyncClient
```

Example of class names in 2.x

```
DynamoDbClient
AcmAsyncClient
```

Region class

The AWS SDK for Java version 1.x had multiple `Region` and `Regions` classes, both in the core package and in many of the service packages. `Region` and `Regions` classes in version 2.x are now collapsed into one core class, `Region`.

Example Region and Regions classes in 1.x

```
com.amazonaws.regions.Region
com.amazonaws.regions.Regions
com.amazonaws.services.ec2.model.Region
```

Example Region class in 2.x

```
software.amazon.awssdk.regions.Region
```

For more details about changes related to using the `Region` class, see Region class name changes (p. 28).

Immutable POJOs

Clients and operation request and response objects are now immutable and cannot be changed after creation. To reuse a request or response variable, you must build a new object to assign to it.

Example of updating a request object in 1.x

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

Example of updating a request object in 2.x

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
        .nextToken(response.nextToken())
        .build();
```

Streaming operations

Streaming operations such as the Amazon S3 `getObject` and `putObject` methods now support non-blocking I/O. As a result, the request and response POJOs no longer take `InputStream` as a parameter.

Instead, the request object accepts `RequestBody`, which is a stream of bytes. The asynchronous client accepts `AsyncRequestBody`.

Example of Amazon `S3putObject` operation in 1.x

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

Example of Amazon `S3putObject` operation in 2.x

```
s3client.putObject(PutObjectRequest.builder()
                                .bucket(BUCKET)
                                .key(KEY)
                                .build(),
                RequestBody.of(Paths.get("myfile.in")));
```

In parallel, the response object accepts `ResponseTransformer` for synchronous clients and `AsyncResponseTransformer` for asynchronous clients.

Example of Amazon `S3getObject` operation in 1.x

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

Example of Amazon `S3getObject` operation in 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
  ResponseTransformer.toFile(Paths.get("key")));
```

Exception changes

Exception class names, their structures, and their relationships have also changed. `software.amazon.awssdk.core.exception.SdkException` is the new base `Exception` class that all the other exceptions extend.

For a full list of the 2.x exception class names mapped to the 1.x exceptions, see Exception class name changes (p. 29).

Service-specific changes

Amazon S3 operation name changes

Many of the operation names for the Amazon S3 client have changed in the AWS SDK for Java 2.x. In version 1.x, the Amazon S3 client is not generated directly from the service API. This results in inconsistency between the SDK operations and the service API. In version 2.x, we now generate the Amazon S3 client to be more consistent with the service API.

Example of Amazon S3 client operation in 1.x

```
changeObjectStorageClass
```

Example of Amazon S3 client operation in 2.x

```
copyObject
```

Example of Amazon S3 client operation in the Amazon S3 service API

```
CopyObject
```

For a full list of the operation name mappings, see the AWS SDK for Java 2.x changelog.

Cross-Region access

For security best practices, cross-Region access is no longer supported for single clients.

In version 1.x, services such as Amazon S3, Amazon SNS, and Amazon SQS are allowed to access resources across Region boundaries. This is no longer allowed in version 2.x using the same client. If you need to access a resource in a different Region, you must create a client in that Region and retrieve the resource using the appropriate client.

Additional client changes

This topic describes additional changes to the default client in the AWS SDK for Java 2.x.

Default client changes

- The default credential provider chain for Amazon S3 no longer includes anonymous credentials. You must specify anonymous access to Amazon S3 manually by using the `AnonymousCredentialsProvider`.
- The following environment variables related to default client creation have been changed.

1.x	2.x
`AWS_CBOR_DISABLED`	`CBOR_ENABLED`
`AWS_ION_BINARY_DISABLE`	`BINARY_ION_ENABLED`

- The following system properties related to default client creation have been changed.

1.x	2.x
`com.amazonaws.sdk.disableEc2Metadata`	`aws.disableEc2Metadata`
`com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	`aws.ec2MetadataServiceEndpoint`
`com.amazonaws.sdk.disableCbor`	`aws.cborEnabled`
`com.amazonaws.sdk.disableIonBinary`	`aws.binaryIonEnabled`

- The following system properties are no longer supported in 2.x.

1.x
`com.amazonaws.sdk.disableCertChecking`
`com.amazonaws.sdk.enableDefaultMetrics`
`com.amazonaws.sdk.enableThrottledRetry`
`com.amazonaws.regions.RegionUtils.fileOverride`

1.x
`com.amazonaws.regions.RegionUtils.disableRemote`
`com.amazonaws.services.s3.disableImplicitGlobalClients`
`com.amazonaws.sdk.enableInRegionOptimizedMode`

- Loading Region configuration from a custom `endpoints.json` file is no longer supported.

Credentials provider changes

Credentials provider

This section provides a mapping of the name changes of credential provider classes and methods between versions 1.x and 2.x of the AWS SDK for Java. The following also lists some of the key differences in the way credentials are processed by the SDK in version 2.x:

- The default credentials provider loads system properties before environment variables in version 2.x. For more information, see Using credentials (p. 35).
- The constructor method is replaced with the `create` or `builder` methods.

  Example

```
DefaultCredentialsProvider.create();
```

- Asynchronous refresh is no longer set by default. You must specify it with the `builder` of the credentials provider.

  Example

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
          .asyncCredentialUpdateEnabled(true)
          .build();
```

- You can specify a path to a custom profile file using the `ProfileCredentialsProvier.builder()`.

  Example

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()

 .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
          .build();
```

- Profile file format has changed to more closely match the AWS CLI. For details, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

Credentials provider changes mapped between versions 1.x and 2.x

Method name changes

1.x	2.x
`AWSCredentialsProvider.getCredentials`	`AwsCredentialsProvider.resolveCredentials`

1.x	2.x
`DefaultAWSCredentialsProviderChain.getInstance`	Not supported
`AWSCredentialsProvider.getInstance`	Not supported
`AWSCredentialsProvider.refresh`	Not supported

Environment variable name changes

1.x	2.x
`AWS_ACCESS_KEY`	`AWS_ACCESS_KEY_ID`
`AWS_SECRET_KEY`	`AWS_SECRET_ACCESS_KEY`
`AWS_CREDENTIAL_PROFILES_FILE`	`AWS_SHARED_CREDENTIALS_FILE`

System property name changes

1.x	2.x
`aws.secretKey`	`aws.secretAccessKey`
`com.amazonaws.sdk.disableEc2Metadata`	`aws.disableEc2Metadata`
`com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	`aws.ec2MetadataServiceEndpoint`

Region class name changes

This section describes the changes implemented in the AWS SDK for Java 2.x for using the `Region` and `Regions` classes.

Region configuration

- Some AWS services don't have Region specific endpoints. When using those services, you must set the Region as `Region.AWS_GLOBAL` or `Region.AWS_CN_GLOBAL`.

  Example

  ```
  Region region = Region.AWS_GLOBAL;
  ```

- `com.amazonaws.regions.Regions` and `com.amazonaws.regions.Region` classes are now combined into one class, `software.amazon.awssdk.regions.Region`.

Method and class name mappings

The following tables map Region related classes between versions 1.x and 2.x of the AWS SDK for Java. You can create an instance of these classes using the `of()` method.

Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

Regions class method changes

1.x	2.x
Regions.fromName	Region.of
Regions.getName	Region.id
Regions.getDescription	Not Supported
Regions.getCurrentRegion	Not Supported
Regions.DEFAULT_REGION	Not Supported
Regions.name	Not Supported

Region class method changes

1.x	2.x
Region.getName	Region.id
Region.hasHttpsEndpoint	Not Supported
Region.hasHttpEndpoint	Not Supported
Region.getAvailableEndpoints	Not Supported
Region.createClient	Not Supported

RegionMetadata class method changes

1.x	2.x
RegionMetadata.getName	RegionMetadata.name
RegionMetadata.getDomain	RegionMetadata.domain
RegionMetadata.getPartition	RegionMetadata.partition

ServiceMetadata class method changes

1.x	2.x
Region.getServiceEndpoint	ServiceMetadata.endpointFor(Region)
Region.isServiceSupported	ServiceMetadata.regions().contains(Region)

Exception class name changes

This topic contains a mapping of exception class-related name changes between versions 1.x and 2.x.

This table maps the exception class name changes.

1.x	2.x
`com.amazonaws.SdkBaseException` `com.amazonaws.AmazonClientException`	`software.amazon.awssdk.core.exception.SdkExceptio`
`com.amazonaws.SdkClientException`	`software.amazon.awssdk.core.exception.SdkClientEx`
`com.amazonaws.AmazonServiceException`	`software.amazon.awssdk.awscore.exception.AwsServi`

The following table maps the methods on exception classes between version 1.x and 2.x.

1.x	2.x
`AmazonServiceException.getRequestId`	`SdkServiceException.requestId`
`AmazonServiceException.getServiceName`	`AwsServiceException.awsErrorDetails().serviceName`
`AmazonServiceException.getErrorCode`	`AwsServiceException.awsErrorDetails().errorCode`
`AmazonServiceException.getErrorMessage`	`AwsServiceException.awsErrorDetails().errorMessag`
`AmazonServiceException.getStatusCode`	`AwsServiceException.awsErrorDetails().sdkHttpResp`
`AmazonServiceException.getHttpHeaders`	`AwsServiceException.awsErrorDetails().sdkHttpResp`
`AmazonServiceException.rawResponse`	`AwsServiceException.awsErrorDetails().rawResponse`

Use the SDK for Java 1.x and 2.x side-by-side

You can use both versions of the AWS SDK for Java in your projects.

The following shows an example of the `pom.xml` file for a project that uses Amazon S3 from version 1.x and DynamoDB from version 2.16.1.

Example Example of POM

This example shows a `pom.xml` file entry for a project that uses both 1.x and 2.x versions of the SDK.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-bom</artifactId>
            <version>1.12.1</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>bom</artifactId>
          <version>2.16.1</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

```
<dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-s3</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```

Use the AWS SDK for Java 2.x

After completing the steps in Setting up the SDK (p. 11), you are ready to make requests to AWS services such as Amazon S3, DynamoDB, IAM, Amazon EC2, and more.

Creating service clients

To make a request to an AWS service, you must first instantiate an object to serve as a client for that service using the static factory method `builder`. Then customize it by using the setters in the builder. The fluent setter methods return the `builder` object, so that you can chain the method calls for convenience and for more readable code. After you configure the properties you want, you can call the `build` method to create the client.

As an example, this code snippet instantiates an `Ec2Client` object as a service client for Amazon EC2:

```
Region region = Region.US_WEST_2;
Ec2Client ec2Client = Ec2Client.builder()
        .region(region)
        .build();
```

Note

Service clients in the SDK are thread-safe. For best performance, treat them as long-lived objects. Each client has its own connection pool resource that is released when the client is garbage collected.

A service client object is immutable, so you must create a new client for each service to which you make requests, or if you want to use a different configuration for making requests to the same service.

Specifying the `Region` in the service client builder is not required for all AWS services; however, it is a best practice to set the Region for the API calls you make in your applications. See AWS region selection (p. 45) for more information.

Using the default client

The client builders have another factory method named `create`. This method creates a service client with the default configuration. It uses the default provider chain to load credentials and the AWS Region. If credentials or the region can't be determined from the environment that the application is running in, the call to `create` fails. See Using credentials (p. 35) and Region selection (p. 45) for more information about how credentials and region are determined.

As an example, this code snippet instantiates a `DynamoDbClient` object as a service client for Amazon DynamoDB:

```
DynamoDbClient dynamoDbClient = DynamoDbClient.create();
```

Making requests

You use the service client to make requests to that AWS service.

For example, this code snippet shows how to create a `RunInstancesRequest` object to create a new Amazon EC2 instance:

```
RunInstancesRequest runInstancesRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

ec2Client.runInstances(runInstancesRequest);
```

Handling responses

You use a response handler to process the response back from the AWS service.

For example, this code snippet shows how to create a `RunInstancesResponse` object to handle the response from Amazon EC2 by printing out the `instanceId` for the new instance from the request above:

```
RunInstancesResponse runInstancesResponse = ec2Client.runInstances(runInstancesRequest);
System.out.println(runInstancesResponse.instances().get(0).instanceId());
```

Closing the client

When you no longer need the service client, close it.

```
ec2Client.close();
```

> **Note**
> Service clients extend the `AutoClosable` interface, but as a best practice - especially with short-lived code such as AWS Lambda functions - you should explicitly call the `close()` method.

Handling exceptions

The SDK uses runtime (or unchecked) exceptions, providing you fine-grained control over error handling and ensuring that exception handling will scale with your application.

An `SdkServiceException`, or one of its sub-classes, is the most common form of exception the SDK will throw. These exceptions represent responses from the AWS service. You can also handle an `SdkClientException`, which occurs when there's a problem on the client side (i.e., in your development or application environment), such a network connection failure.

This code snippet demonstrates one way to handle service exceptions while uploading a file to Amazon S3.

```
Region region = Region.US_WEST_2;
s3Client = S3Client.builder()
        .region(region)
        .build();

try {
```

```
    PutObjectRequest putObjectRequest = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .build();

    s3Client.putObject(putObjectRequest, RequestBody.fromString("SDK for Java test"));

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

See Handling exceptions (p. 66) for more information.

Using waiters

Some requests take time to process, such as creating a new table in DynamoDB or creating a new Amazon S3 bucket. To ensure the resource is ready before your code continues to run, use a *Waiter*.

For example, this code snippet creates a new table ("myTable") in DynamoDB, waits for the table to be in an `ACTIVE` status, and then prints out the response:

```
DynamoDbClient dynamoDbClient = DynamoDbClient.create();
DynamoDbWaiter dynamoDbWaiter = dynamoDbClient.waiter();

WaiterResponse<DescribeTableResponse> waiterResponse =
   dynamoDbWaiter.waitUntilTableExists(r -> r.tableName("myTable"));

waiterResponse.matched().response().ifPresent(System.out::println);
```

See Using waiters (p. 101) for more information.

Configuring service clients

To customize the configuration of a service client, use the setters on the factory method `builder`. For convenience and to create more readable code, you chain the methods to set multiple configuration options.

As an example, refere to the following code snippet.

```
ClientOverrideConfiguration clientOverrideConfiguration =
    ClientOverrideConfiguration.builder()
                    .apiCallAttemptTimeout(Duration.ofSeconds(1))
                    .retryPolicy(RetryPolicy.builder().numRetries(10).build())
                    .addMetricPublisher(CloudWatchMetricPublisher.create())
                    .build();

Region region = Region.US_WEST_2;
S3Client s3Client = S3Client.builder()
                    .region(region)
                    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                    .overrideConfiguration(clientOverrideConfiguration)
                    .httpClientBuilder(ApacheHttpClient.builder()

 .proxyConfiguration(proxyConfig.build(ProxyConfiguration.builder()))
```

```
                        .build())
                    .build();
```

HTTP clients

You can change the default configuration for HTTP clients in applications you build with the AWS SDK for Java. For information on how to configure HTTP clients and settings, see HTTP configuration (p. 50).

Retries

You can change the default settings for retries in your service clients, including the retry mode and back-off strategy. For more information, refer to the `RetryPolicy` class in the AWS SDK for Java API Reference.

For more information about retries in AWS services, see Error retries and exponential backoff in AWS.

Timeouts

You can configure timeouts for each of your service clients using the `apiCallTimeout` and the `apiCallAttemptTimeout` setters. The `apiCallTimeout` setting is the amount of time to allow the client to complete the execution of an API call. The `apiCallAttemptTimeout` setting is the amount of time to wait for the HTTP request to complete before giving up.

For more information, see `apiCallTimeout` and `apiCallAttemptTimeout` in the AWS SDK for Java API Reference.

Execution interceptors

You can write code that intercepts the execution of your API requests and responses at different parts of the request/response lifecycle. This enables you to publish metrics, modify a request in-flight, debug request processing, view exceptions, and more. For more information, see the `ExecutionInterceptor` interface in the AWS SDK for Java API Reference.

Additional information

- For complete examples of the code snippets above, see Working with Amazon DynamoDB (p. 120), Working with Amazon EC2 (p. 141), and Working with Amazon S3 (p. 106).

Provide temporary credentials to the SDK

Before making a request to Amazon Web Services using the AWS SDK for Java 2.x, the SDK cryptographically signs temporary credentials issued by AWS. To access temporary credentials, the SDK retrieves configuration values by checking several locations.

This topic discusses several ways that you enable to SDK to access temporary credentials.

Topics

- Use temporary credentials (p. 36)
- Default credentials provider chain (p. 37)
- Use a specific credentials provider or provider chain (p. 38)
- Use profiles (p. 38)
- Load temporary credentials from an external process (p. 40)
- Supply temporary credentials in code (p. 43)
- Configuring IAM roles for Amazon EC2 (p. 44)

Use temporary credentials

For increased security, AWS recommends that you configure the SDK for Java to use temporary credentials instead of long-lived credentials. Temporary credentials consist of access keys (access key id and secret access key) and a session token. We recommend that you configure the SDK (p. 36) to automatically get temporary credentials, since the token refresh process is automatic. You can, however, provide the SDK with temporary credentials (p. 36) directly.

IAM Identity Center configuration

When you configure the SDK to use IAM Identity Center single sign-on access as described in the section called "Basic setup" (p. 12) in this guide, the SDK automatically uses temporary credentials.

The SDK uses the IAM Identity Center access token to gain access to the IAM role that is configured with the `sso_role_name` setting in your `config` file. The SDK assumes this IAM role and retrieves temporary credentials to use for AWS service requests.

For more details about how the SDK gets temporary credentials from the configuration, see the Understanding IAM Identity Center authentication section of the AWS SDKs and Tools Reference Guide.

Retrieve from AWS access portal

As an alternative to IAM Identity Center single sign-on configuration, you can copy and use temporary credentials available in the AWS access portal. You can use the temporary credentials in a profile or use them as values for system properties and environment variables.

Set up a local credentials file for temporary credentials

1. Create a shared credentials file
2. In the credentials file, paste the following placeholder text until you paste in working temporary credentials.

```
[default]
aws_access_key_id=<value from AWS access portal>
aws_secret_access_key=<value from AWS access portal>
aws_session_token=<value from AWS access portal>
```

3. Save the file. The file `~/.aws/credentials` should now exist on your local development system. This file contains the [default] profile that the SDK for Java uses if a specific named profile is not specified.
4. Sign in to the AWS access portal

5. Follow these instructions to copy IAM role credentials from the AWS access portal.

   Note
   Although the title of the page reads *Getting IAM role credentials for CLI access*, the instructions apply to the SDK for Java and also the AWS Command Line Interface (AWS CLI).

   a. For step 2 in the instructions, choose the AWS account and IAM role name that grants access for your development needs. This role typically has a name like **PowerUser** or **Developer**.
   b. For step 4, select the option that reads **Add a profile to your AWS credentials file** and copy the contents.
6. Paste the copied credentials into your local `credentials` file and remove the generated profile name. Your file should resemble the following.

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGSTRINGEX
```

7. Save the `credentials` file.

When the SDK for Java creates a service client, it will access these temporary credentials and use them for each request. The settings for the IAM role chosen in step 5a determine how long the temporary credentials are valid. The maximum duration is twelve hours.

After the temporary credentials expire, repeat steps 4 through 7.

Default credentials provider chain

The default credentials provider chain is implemented by the DefaultCredentialsProvider class. It sequentially checks each place where you can set the default configuration for supplying temporary credentials, and then selects the first one you set.

To use the default credentials provider chain to supply temporary credentials that are used in your application, create a service client builder but don't specify a credentials provider. The following code snippet creates a `DynamoDbEnhancedClient` that uses the default credentials provider chain to locate and retrieve default configuration settings.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb =
    DynamoDbEnhancedClient.builder()
                          .region(region)
                          .build();
```

Credential settings retrieval order

The default credentials provider chain of the SDK for Java 2.x searches for configuration in your environment using a predefined sequence.

1. Java system properties
   - The SDK uses the SystemPropertyCredentialsProvider class to load temporary credentials from the `aws.accessKeyId`, `aws.secretAccessKey`, and `aws.sessionToken` Java system properties.

     Note
     For information on how to set Java system properties, see the System Properties tutorial on the official *Java Tutorials* website.
2. Environment variables

- The SDK uses the EnvironmentVariableCredentialsProvider class to load temporary credentials from the `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` environment variables.

3. Web identity token from AWS Security Token Service
   - The SDK uses the WebIdentityTokenFileCredentialsProvider class to load temporary credentials from Java system properties or environment variables.
4. The shared `credentials` and `config` files
   - The SDK uses the ProfileCredentialsProvider to load IAM Identity Center single sign-on settings or temporary credentials from the `[default]` profile in the shared `credentials` and `config` files.

     The AWS SDKs and Tools Reference Guide has detailed information about how the SDK for Java works with the IAM Identity Center single sign-on token to get temporary credentials that the SDK uses to call AWS services.

     > **Note**
     > The `credentials` and `config` files are shared by various AWS SDKs and Tools. For more information, see The .aws/credentials and .aws/config files in the AWS SDKs and Tools Reference Guide.
5. Amazon ECS container credentials
   - The SDK uses the ContainerCredentialsProvider class to load temporary credentials from the `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` system environment variable.
6. Amazon EC2 instance IAM role-provided credentials
   - The SDK uses the InstanceProfileCredentialsProvider class to load temporary credentials from the Amazon EC2 metadata service.

Use a specific credentials provider or provider chain

As an alternative to the default credentials provider chain, you can specify which credentials provider the SDK should use. When you supply a specific credentials provider, the SDK skips the process of checking various locations, which slightly reduces the time to create a service client.

For example, if you set your default configuration using environment variables, supply an EnvironmentVariableCredentialsProvider object to the `credentialsProvider` method on the service client builder, as in the following code snippet.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb = DynamoDbClient.builder()
      .region(region)
      .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
      .build();
```

For a complete list of credential providers and provider chains, see **All Known Implementing Classes** in AwsCredentialsProvider.

> **Note**
> You can use your own credentials provider or provider chains by implementing the `AwsCredentialsProvider` interface.

Use profiles

Using the shared `config` and `credentials` file, you can set up several profiles. This enables your application to use multiple sets of credentials configuration. The `[default]` profile was mentioned previously. The SDK uses the ProfileCredentialsProvider class to load settings from profiles defined in the shared `credentials` file.

The following code snippet demonstrates how to build a service client that uses the settings defined as part of the profile named `my_profile`.

```
Region region = Region.US_WEST_2;
DynamoDbClient ddb = DynamoDbClient.builder()
      .region(region)
      .credentialsProvider(ProfileCredentialsProvider.create("my_profile"))
      .build();
```

Set a different profile as the default

To set a profile other than the `[default]` profile as the default for your application, set the `AWS_PROFILE` environment variable to the name of your custom profile.

To set this variable on Linux, macOS, or Unix, use `export`:

```
export AWS_PROFILE="other_profile"
```

To set these variables on Windows, use `set`:

```
set AWS_PROFILE="other_profile"
```

Alternatively, set the `aws.profile` Java system property to the name of the profile.

Reload profile credentials

You can configure any credentials provider that has a `profileFile()` method on its builder to reload profile credentials. These credentials profile classes are: `ProfileCredentialsProvider`, `DefaultCredentialsProvider`, `InstanceProfileCredentialsProvider`, and `ProfileTokenProvider`.

> **Note**
> Profile credential reloading works only with the following settings in the profile file : `aws_access_key_id`, `aws_secret_access_key`, and `aws_session_token`. Settings such as `region`, `sso_session`, `sso_account_id`, and `source_profile` are ignored.

To configure a supported credentials provider to reload profile settings, provide an instance of `ProfileFileSupplier` to the `profileFile()` builder method. The following code example demonstrates a `ProfileCredentialsProvider` that reloads credential settings from the `[default]` profile.

```
ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.defaultSupplier())
    .build();

// Set up a service client with the provider instance.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(provider)
                    .build();

/*
    Before dynamoDbClient makes a request, it reloads the credentials settings
    by calling provider.resolveCredentials().
*/
```

When `ProfileCredentialsProvider.resolveCredentials()` is called, the SDK for Java reloads the settings. `ProfileFileSupplier.defaultSupplier()` is one of several convenience implementations of `ProfileFileSupplier` provided by the SDK. If your use case requires, you can provide your own implementation.

The following example shows the use of the `ProfileFileSupplier.reloadWhenModified()` convenience method. `reloadWhenModified()` takes a `Path` parameter, which gives you flexibility in designating the source file for the configuration rather than the standard `~/.aws/credentials` (or `config`) location.

The settings will be reloaded when `resolveCredentials()` is called only if SDK determines the file's content has been modified.

```
Path credentialsFilePath = ...

ProfileCredentialsProvider provider = ProfileCredentialsProvider
    .builder()
    .profileFile(ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
 ProfileFile.Type.CREDENTIALS))
    .profileName("my-profile")
    .build();
/*
    A service client configured with the provider instance calls
 provider.resolveCredential()
    before each request.
*/
```

The `ProfileFileSupplier.aggregate()` method merges the contents of multiple configuration files. You decide whether a file is reloaded per call to `resolveCredentials()` or a file's settings are fixed at the time it was first read.

The following example shows a `DefaultCredentialsProvider` that merges the settings of two files that contain profile settings. The SDK reloads the settings in the file pointed to by the `credentialsFilePath` variable each time `resolveCredentials()` is called and the settings have changed. The settings from the `profileFile` object remain the same.

```
Path credentialsFilePath = ...;
ProfileFile profileFile = ...;

DefaultCredentialsProvider provider = DefaultCredentialsProvider
        .builder()
        .profileFile(ProfileFileSupplier.aggregate(
                ProfileFileSupplier.reloadWhenModified(credentialsFilePath,
 ProfileFile.Type.CREDENTIALS),
                ProfileFileSupplier.fixedProfileFile(profileFile)))
        .profileName("my-profile")
        .build();
/*
    A service client configured with the provider instance calls
 provider.resolveCredential()
    before each request.
*/
```

Load temporary credentials from an external process

Warning

The following describes a method of sourcing temporary credentials from an external process. This can potentially be dangerous, so proceed with caution. Other credential providers should be preferred if at all possible. If using this option, you should make sure that the `config` file is as locked down as possible using security best practices for your operating system.

Make sure that your custom credentials tool does not write any secret information to `StdErr`. SDKs and AWS CLI can capture and log such information, potentially exposing it to unauthorized users.

With the SDK for Java 2.x, you can acquire temporary credentials from an external process for custom use cases. There are two ways to configure this functionality.

Use the `credential_process` setting

If you have a method that provides temporary credentials, you can integrate it by adding the `credential_process` setting as part of a profile definition in the `config` file. The value you specify must use the full path to the command file. If the file path contains any spaces, you must surround it with quotation marks.

The SDK calls the command exactly as given and then reads JSON data from `stdout`.

The following examples show the use of this setting for file paths without spaces and file paths with spaces.

Linux/macOS

No spaces in file path

```
[profile process-credential-profile]
credential_process = /path/to/credential/file/credential_file.sh --custom-command
 custom_parameter
```

Spaces in file path

```
[profile process-credential-profile]
credential_process = "/path/with/space to/credential/file/credential_file.sh" --
custom-command custom_parameter
```

Windows

No spaces in file path

```
[profile process-credential-profile]
credential_process = C:\Path\To\credentials.cmd --custom_command custom_parameter
```

Spaces in file path

```
[profile process-credential-profile]
credential_process = "C:\Path\With Space To\credentials.cmd" --custom_command
 custom_parameter
```

The following code snippet demonstrates how to build a service client that uses the temporary credentials defined as part of the profile named `process-credential-profile`.

```
Region region = Region.US_WEST_2;
S3Client s3Client = S3Client.builder()
      .region(region)
      .credentialsProvider(ProfileCredentialsProvider.create("process-credential-profile"))
      .build();
```

For detailed information about using an external process as a source of temporary credentials, refer to the process credentials section in the AWS SDKs and Tools Reference Guide.

Use the `ProcessCredentialsProvider`

As an alternative to using settings in the `config` file, you can use the SDK's `ProcessCredentialsProvider` to load temporary credentials using Java.

The following examples show various versions of how to specify an external process using the `ProcessCredentialsProvider` and configuring a service client that uses the temporary credentials.

Linux/macOS

No spaces in file path

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("/path/to/credentials.sh optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
                      .build();
```

Spaces in file path

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("/path\\ with\\ spaces\\ to/credentials.sh optional_param1
 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
                      .build();
```

Windows

No spaces in file path

```
ProcessCredentialsProvider credentials =
    ProcessCredentialsProvider
        .builder()
        .command("C:\\Path\\To\\credentials.exe optional_param1 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
                      .build();
```

Spaces in file path

```
ProcessCredentialsProvider credentials =
```

```
    ProcessCredentialsProvider
        .builder()
        .command("\"C:\\Path\\With Spaces To\\credentials.exe\" optional_param1
 optional_param2")
        .build();

S3Client s3 = S3Client.builder()
                      .region(Region.US_WEST_2)
                      .credentialsProvider(credentials)
                      .build();
```

Supply temporary credentials in code

If the default credential chain or a specific or custom provider or provider chain doesn't work for your application, you can supply temporary credentials directly in code. These can be IAM role credentials as described above (p. 36) or temporary credentials retrieved from AWS Security Token Service (AWS STS). If you retrieved temporary credentials using AWS STS, provide them to an AWS service client as shown in the following code example.

1. Instantiate a class that provides the AwsCredentials interface, such as AwsSessionCredentials. Supply it with the AWS access keys and session token to use for the connection.
2. Create a StaticCredentialsProvider object and supply it with the `AwsSessionCredentials` object.
3. Configure the service client builder with the `StaticCredentialsProvider` and build the client.

The following example creates an Amazon S3 service client using temporary credentials that you supply.

```
public static void assumeRole(String roleArn, String roleSessionName) {
    // The IAM role represented by the roleArn parameter can be assumed by any user in the
 same account
    // and has read-only permissions when it accesses Amazon S3.

    // The default credentials provider chain will find the single sign-on settings in the
 [default] profile.
    StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();
    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials tempRoleCredentials = roleResponse.credentials();
        // The following temporary credential items are used when Amazon S3 is called
        String key = tempRoleCredentials.accessKeyId();
        String secKey = tempRoleCredentials.secretAccessKey();
        String secToken = tempRoleCredentials.sessionToken();

        // List all buckets in the account using the temporary credentials retrieved by
 invoking assumeRole.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()

 .credentialsProvider(StaticCredentialsProvider.create(AwsSessionCredentials.create(key,
 secKey, secToken)))
                .region(region)
                .build();

        List<Bucket> buckets = s3.listBuckets().buckets();
```

```
        for (Bucket bucket : buckets) {
            System.out.println("bucket name: " + bucket.name());
        }
    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Configuring IAM roles for Amazon EC2

You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

This topic provides information about how to use IAM roles with AWS SDK for Java applications running on Amazon EC2.

Default provider chain and Amazon EC2 instance profiles

If your application creates an AWS client using the `create` method, the client searches for temporary credentials using the *default credentials provider chain*, as described in the the section called "Default credentials provider chain" (p. 37) section .

The final step in the default provider chain is available only when running your application on an Amazon EC2 instance. However, it provides better ease of use and security when working with Amazon EC2 instances. You can also pass an InstanceProfileCredentialsProvider instance directly to the client constructor to get instance profile credentials without going through the entire default provider chain.

This is demonstrated in the following example:

```
S3Client s3 = S3Client.builder()
       .credentialsProvider(InstanceProfileCredentialsProvider.builder().build())
       .build();
```

When you use this approach, the SDK retrieves temporary AWS credentials that have the same permissions as those associated with the IAM role that is associated with the Amazon EC2 instance in its instance profile. Although these credentials are temporary and would eventually expire, `InstanceProfileCredentialsProvider` periodically refreshes them for you so that they continue to allow access to AWS.

Walkthrough: Using IAM roles for EC2 instances

This walkthrough shows you how to retrieve an object from Amazon S3 using an IAM role to manage access.

Create an IAM role

Create an IAM role that grants read-only access to Amazon S3.

1. Open the IAM console.
2. In the navigation pane, choose **Roles**, then **Create Role**.

3. On the **Select trusted entity** page, choose **AWS service**. Under **Use case**, choose **EC2**. Choose **Next**.
4. On the **Add permissions** page, choose **Amazon S3ReadOnlyAccess** from the permissions policy list, then choose **Next**.

   Enter a name for the role, then select **Create role**. Note this name. You need it to launch your Amazon EC2 instance.

Launch an EC2 instance and specify your IAM role

To launch an Amazon EC2 instance with an IAM role using the console, follow the instructions in the Amazon EC2 User Guide for Linux Instances.

> **Note**
> You need to create or use an existing security group and key pair to connect to the instance.

With this IAM and Amazon EC2 setup, you can deploy your application to the Amazon EC2 instance and it will have read access to Amazon S3.

AWS region selection

Regions enable you to access AWS services that physically reside in a specific geographic area. This can be useful both for redundancy and to keep your data and applications running close to where you and your users will access them.

In AWS SDK for Java 2.x, all the different region related classes from version 1.x have been collapsed into one Region class. You can use this class for all region-related actions such as retrieving metadata about a region or checking whether a service is available in a region.

Choosing a region

You can specify a region name and the SDK will automatically choose an appropriate endpoint for you.

To explicitly set a region, we recommend that you use the constants defined in the Region class. This is an enumeration of all publicly available regions. To create a client with a region from the class, use the following code.

```
Ec2Client ec2 = Ec2Client.builder()
          .region(Region.US_WEST_2)
          .build();
```

If the region you are attempting to use isn't one of the constants in the `Region` class, you can create a new region using the `of` method. This feature allows you access to new Regions without upgrading the SDK.

```
Region newRegion = Region.of("us-east-42");
Ec2Client ec2 = Ec2Client.builder()
          .region(newRegion)
          .build();
```

> **Note**
> After you build a client with the builder, it's *immutable* and the region *cannot be changed*. If you are working with multiple AWS Regions for the same service, you should create multiple clients —one per region.

Choosing a specific endpoint

Each AWS client can be configured to use a *specific endpoint* within a region by calling the `endpointOverride` method.

For example, to configure the Amazon EC2 client to use the Europe (Ireland) Region, use the following code.

```
Ec2Client ec2 = Ec2Client.builder()
               .region(Region.EU_WEST_1)
               .endpointOverride(URI.create("https://ec2.eu-west-1.amazonaws.com"))
               .build();
```

See Regions and Endpoints for the current list of regions and their corresponding endpoints for all AWS services.

Automatically determine the Region from the environment

When running on Amazon EC2 or AWS Lambda, you might want to configure clients to use the same region that your code is running on. This decouples your code from the environment it's running in and makes it easier to deploy your application to multiple regions for lower latency or redundancy.

To use the default credential/region provider chain to determine the region from the environment, use the client builder's `create` method.

```
Ec2Client ec2 = Ec2Client.create();
```

If you don't explicitly set a region using the `region` method, the SDK consults the default region provider chain to try and determine the region to use.

Default region provider chain

The following is the region lookup process:

1. Any explicit region set by using `region` on the builder itself takes precedence over anything else.
2. The `AWS_REGION` environment variable is checked. If it's set, that region is used to configure the client.

   Note
   This environment variable is set by the Lambda container.
3. The SDK checks the AWS shared configuration file (usually located at `~/.aws/config`). If the *region* property is present, the SDK uses it.
   - The `AWS_CONFIG_FILE` environment variable can be used to customize the location of the shared config file.
   - The `AWS_PROFILE` environment variable or the `aws.profile` system property can be used to customize the profile that the SDK loads.
4. The SDK attempts to use the Amazon EC2 instance metadata service to determine the region of the currently running Amazon EC2 instance.
5. If the SDK still hasn't found a region by this point, client creation fails with an exception.

When developing AWS applications, a common approach is to use the *shared configuration file* (described in Credential retrieval order (p. 37)) to set the region for local development, and rely on the default

region provider chain to determine the region when running on AWS infrastructure. This greatly simplifies client creation and keeps your application portable.

Checking for service availability in a Region

To see if a particular AWS service is available in a region, use the `serviceMetadata` and `region` method on the service that you'd like to check.

```
DynamoDbClient.serviceMetadata().regions().forEach(System.out::println);
```

See the Region class documentation for the regions you can specify, and use the endpoint prefix of the service to query.

Reducing SDK startup time for AWS Lambda

One of the goals of the AWS SDK for Java 2.x is to reduce the startup latency for AWS Lambda functions. The SDK contains changes that reduce startup time, which are discussed at the end of this topic.

First, this topic focuses on changes that you can make to reduce cold start times. These include making changes in your code structure and in the configuration of service clients.

Use the SDK's `UrlConnectionHttpClient`

For *synchronous* scenarios, the SDK for Java 2.x offers the `UrlConnectionHttpClient` class, which is based on the JDK's HTTP client classes. Because the `UrlConnectionHttpClient` is based on classes already on the classpath, there are no extra dependencies to load.

For information on adding the `UrlConnectionHttpClient` to your Lambda project and configuring its use, see Configuring the URLConnection-based HTTP client (p. 55).

> **Note**
> There are some feature limitations with the `UrlConnectionHttpClient` in comparison to the SDK's `ApacheHttpClient`. The `ApacheHttpClient` is the default asynchronous HTTP client in the SDK. For example, the `UrlConnectionHttpClient` does not support the HTTP PATCH method.
> A handful of AWS API operations require PATCH requests. Those operation names usually start with `Update*`. The following are several examples.
>
> - Several `Update*` operations in the AWS Security Hub API and also the BatchUpdateFindings operation
> - All Amazon API Gateway API `Update*` operations
> - Several `Update*` operations in the Amazon WorkDocs API
>
> If you might use the `UrlConnectionHttpClient`, first refer to the API Reference for the AWS service that you're using. Check to see if the operations you need use the PATCH operation.

Use the SDK's `AwsCrtAsyncHttpClient`

The `AwsCrtAsyncHttpClient` is the *asynchronous* counterpart for reducing Lambda startup time in the SDK.

The `AwsCrtAsyncHttpClient` is an asynchronous, non-blocking HTTP client. It's built on top of the Java bindings of the AWS Common Runtime, which is written in the C programming language. Among the goals in the development of the AWS Common Runtime is fast performance.

This guide's section on configuring HTTP clients (p. 62) has information about adding an `AwsCrtAsyncHttpClient` to your Lambda project and configuring its use.

Remove unused HTTP client dependencies

Along with the explicit use of `UrlConnectionHttpClient` or `AwsCrtAsyncHttpClient`, you can remove other HTTP clients that the SDK brings in by default. Lambda startup time is reduced when fewer libraries need to be loaded, so you should remove any unused artifacts that the JVM needs to load.

The following snippet of a Maven `pom.xml` file shows the exclusion of the Apache-based HTTP client and the Netty-based HTTP client. (These clients aren't needed when you use the `UrlConnectionHttpClient`.) This example excludes the HTTP client artifacts from the S3 client dependency and adds the `url-connection-client` artifact, which brings in the `UrlConnectionHttpClient` class.

```
<project>
    <properties>
        <aws.java.sdk.version>2.17.290</aws.java.sdk.version>
    <properties>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.java.sdk.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>url-connection-client</artifactId>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>netty-nio-client</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>software.amazon.awssdk</groupId>
                    <artifactId>apache-client</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>
</project>
```

If you use the `AwsCrtAsyncHttpClient`, replace the dependency on the `url-connection-client` to a dependency on the `aws-crt-client`.

Note
Add the `<exclusions>` element to all service client dependencies in your `pom.xml` file.

Configure service clients to shortcut lookups

Specify a region

When you create a service client, call the `region` method on the service client builder. This shortcuts the SDK's default Region lookup process that checks several places for the AWS Region information.

To keep the Lambda code independent of the region, use the following code inside the `region` method. This code accesses the `AWS_REGION` environment variable set by the Lambda container.

```
Region.of(System.getenv(SdkSystemSetting.AWS_REGION.environmentVariable()))
```

Use the `EnvironmentVariableCredentialProvider`

Much like the default lookup behavior for the Region information, the SDK looks in several places for credentials. By specifying the `EnvironmentVariableCredentialProvider` when you build a service client, you save time in the SDK's lookup process.

> **Note**
> Using this credentials provider enables the code to be used in Lambda functions, but might not work on Amazon EC2 or other systems.

The following code snippet shows an S3 service client appropriately configured for use in a Lambda environment.

```
S3Client client = S3Client.builder()
       .region(Region.of(System.getenv(SdkSystemSetting.AWS_REGION.environmentVariable())))
       .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
       .httpClient(UrlConnectionHttpClient.builder().build())
       .build();
```

Initialize the SDK client outside of the Lambda function handler

We recommend initializing an SDK client outside of the Lambda handler method. This way, if the execution context is reused, the initialization of the service client can be skipped. By reusing the client instance and its connections, subsequent invocations of the handler method occur more quickly.

In the following example, the `S3Client` instance is initialized in the constructor using a static factory method. If the container that is managed by the Lambda environment is reused, the initialized `S3Client` instance is reused.

```
public class App implements RequestHandler<Object, Object> {
    private final S3Client s3Client;

    public App() {
        s3Client = DependencyFactory.s3Client();
    }

    @Override
    public Object handle Request(final Object input, final Context context) {
         ListBucketResponse response = s3Client.listBuckets();
         // Process the response.
    }
}
```

Minimize dependency injection

Dependency injection (DI) frameworks might take additional time to complete the setup process. They might also require additional dependencies, which take time to load.

If a DI framework is needed, we recommend using lightweight DI frameworks such as Dagger.

Use a Maven Archetype targeting AWS Lambda

The AWS Java SDK team has developed a Maven Archetype template to bootstrap a Lambda project with minimal startup time. You can build out a Maven project from the archetype and know that the dependencies are configured suitably for the Lambda environment.

To learn more about the archetype and work through an example deployment, see this blog post.

Consider Lambda SnapStart for Java

If your runtime requirements are compatible, AWS offers Lambda SnapStart for Java. Lambda SnapStart is an infrastructure-based solution that improves startup performance for Java functions. When you publish a new version of a function, Lambda SnapStart initializes it and takes an immutable, encrypted snapshot of the memory and disk state. SnapStart then caches the snapshot for reuse.

Version 2.x changes that affect startup time

In addition to changes that you make to your code, version 2.x of the SDK for Java includes three primary changes that reduce startup time:

- Use of jackson-jr, which is a serialization library that improves initialization time
- Use of the java.time libraries for date and time objects, which is part of the JDK
- Use of Slf4j for a logging facade

Additional resources

The AWS Lambda Developer Guide contains a section on best practices for developing Lambda functions that is not Java specific.

For an example of building a cloud-native application in Java that uses AWS Lambda, see this workshop content. The workshop discussion performance optimization and other best practices.

You can consider using static images that are compiled ahead of time to reduce startup latency. For example, you can use the SDK for Java 2.x and Maven to build a GraalVM native image (p. 19).

Configuring HTTP clients

HTTP clients in the SDK

You can change the default configuration for HTTP clients in applications you build with the AWS SDK for Java 2.x. This section discusses HTTP clients and settings for the SDK.

Synchronous service clients

A synchronous service client, such as the S3Client or the DynamoDbClient, requires the use of a synchronous HTTP client. The AWS SDK for Java offers two synchronous HTTP clients.

ApacheHttpClient (default)

ApacheHttpClient is the default HTTP client for synchronous service clients. For information about configuring the ApacheHttpClient, see Configuring the Apache-based HTTP client (p. 52).

UrlConnectionHttpClient

As a lighter weight option to the ApacheHttpClient, you can use the UrlConnectionHttpClient. For information about configuring the UrlConnectionHttpClient, see Configuring the URLConnection-based HTTP client (p. 55).

Asynchronous service clients

An asynchronous service client, such as the S3AsyncClient or the DynamoDbAsyncClient, requires the use of an asynchronous HTTP client. The AWS SDK for Java offers two asynchronous HTTP clients.

NettyNioAsyncHttpClient (default)

NettyNioAsyncHttpClient is the default HTTP client used by asynchronous clients. For information about configuring the NettyNioAsyncHttpClient, see the section called “Configuring the Netty-based HTTP client” (p. 59).

AwsCrtAsyncHttpClient

The new AwsCrtAsyncHttpClient, which also has a quicker loading time compared to the NettyNioAsyncHttpClient, is also available. For information about configuring the AwsCrtAsyncHttpClient, see the section called “Configuring the AWS CRT-based HTTP client” (p. 62).

Smart configuration defaults

The AWS SDK for Java 2.x (version 2.17.102 or later) offers a smart configuration defaults feature. This feature optimizes two HTTP client properties along with other properties that don't affect the HTTP client.

The smart configuration defaults set sensible values for the connectTimeoutInMillis and tlsNegotiationTimeoutInMillis properties based on a defaults mode value that you provide. You choose the defaults mode value based on your application's characteristics.

For more information about smart configuration defaults and how to choose the defaults mode value that is best suited for your applications, see the AWS SDKs and Tools Reference Guide.

Following are four ways to set the defaults mode for your application.

Service client

Use the service client builder to configure the defaults mode directly on the service client. The following example sets the defaults mode to auto for the DynamoDbClient.

```
DynamoDbClient ddbClient = DynamoDbClient.builder()
                            .defaultsMode(DefaultsMode.AUTO)
                            .build();
```

System property

You can use the `aws.defaultsMode` system property to specify the defaults mode. If you set the system property in Java, you need to set the property before initializing any service client.

The following example shows you how to set the defaults mode to `auto` using a system property set in Java.

```
System.setProperty("aws.defaultsMode", "auto");
```

The following example demonstrates how you set the defaults mode to `auto` using a `-D` option of the `java` command.

```
java -Daws.defaultsMode=auto
```

Environment variable

Set a value for environment variable `AWS_DEFAULTS_MODE` to select the defaults mode for your application.

The following information shows the command to run to set the value for the defaults mode to `auto` using an environment variable.

Operating system	Command to set environment variables
Linux, macOS, or Unix	`export AWS_DEFAULTS_MODE=auto`
Windows	`set AWS_DEFAULTS_MODE=auto`

AWS config file

You can add a `defaults_mode` configuration property to the shared AWS `config` file as the following example shows.

```
[default]
defaults_mode = auto
```

Configuring the Apache-based HTTP client

Synchronous service clients in the AWS SDK for Java 2.x use an Apache-based HTTP client, ApacheHttpClient by default. The SDK's `ApacheHttpClient` is based on the Apache HttpClient.

The SDK also offers the UrlConnectionHttpClient, which loads more quickly, but has fewer features. For information about configuring the `UrlConnectionHttpClient`, see the section called "Configuring the URLConnection-based HTTP client" (p. 55).

To see the full set of configuration options available to you for the `ApacheHttpClient`, see ApacheHttpClient.Builder and ProxyConfiguration.Builder.

Accessing the `ApacheHttpClient`

In most situations, you use the `ApacheHttpClient` without any explicit configuration. You declare your service clients and the SDK will configure the ApacheHttpClient with standard values for you.

If you want to explicitly configure the `ApacheHttpClient` or use it with multiple service clients, you need to make it available for configuration.

No configuration needed

When you declare a dependency on a service client in Maven, the SDK adds a *runtime* dependency on the `apache-client` artifact. This makes the `ApacheHttpClient` class available to your code at runtime. If you are not configuring the Apache-based HTTP client, you do not need to specify a dependency for it.

In the following XML snippet of a Maven `pom.xml` file, the dependency declared with `<artifactId>s3</artifactId>` automatically brings in the Apache-based HTTP client. You don't need to declare a dependency specifically for it.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.17.290</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <!-- The s3 dependency automatically adds a runtime dependency on the
 ApacheHttpClient-->
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>
</dependencies>
```

With these dependencies, you cannot make any explicit HTTP configuration changes, because the `ApacheHttpClient` library is only on the runtime classpath.

Configuration needed

To configure the `ApacheHttpClient`, you need to add a dependency on the `apache-client` library at *compile* time.

Refer to the following example of a Maven `pom.xml` file to configure the `ApacheHttpClient`.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.17.290</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
        </dependency>
        <!-- By adding the apache-client dependency, ApacheHttpClient will be added to
             the compile classpath so you can configure it. -->
        <dependency>
```

```
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>apache-client</artifactId>
        </dependency>
    </dependencies>
```

Configuring the ApacheHttpClient

You can configure an instance of `ApacheHttpClient` along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use the `ApacheHttpClient.Builder` to configure the properties for the Apache-based HTTP client.

Best practice: dedicate an ApacheHttpClient instance to a service client

If you need to configure an instance of the `ApacheHttpClient`, we recommend that you build the dedicated `ApacheHttpClient` instance. You can do so by using the `httpClientBuilder` method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the `ApacheHttpClient` instance is not closed down when it's no longer needed.

The following example creates an `S3Client` and configures the embedded instance of `ApacheHttpClient` with `maxConnections` and `connectionTimeout` values. The HTTP instance is created using the `httpClientBuilder` method of `S3Client.Builder`.

Imports

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Code

```
S3Client s3Client = S3Client   // Singleton: Use the s3Client for all requests.
    .builder()
    .httpClientBuilder(ApacheHttpClient.builder()
        .maxConnections(100)
        .connectionTimeout(Duration.ofSeconds(5))
    ).build();

// Perform work with the s3Client.

s3Client.close();   // Requests completed: Close all service clients.
```

Alternative approach: share an ApacheHttpClient instance

To help keep resource and memory usage lower for your application, you can configure an `ApacheHttpClient` and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
> When an `ApacheHttpClient` instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures an Apache-based HTTP client, which is used by two service clients. The configured `ApacheHttpClient` instance is passed to the `httpClient` method of the service client's builder. When the service clients and the HTTP client are no longer needed, they are explicitly closed. The HTTP client is closed last.

Imports

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

Code

```
SdkHttpClient apacheHttpClient =
    ApacheHttpClient.builder()
                    .maxConnections(100)
                    .tcpKeepAlive(true)
                    .build();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client =
    S3Client.builder()
            .httpClient(apacheHttpClient).build();

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
                   .httpClient(apacheHttpClient).build();

// Perform work with the s3Client and dynamoDbClient.

// Requests completed: Close all service clients.
s3Client.close();
dynamoDbClient.close();
apacheHttpClient.close();  // Explicitly close apacheHttpClient.
```

Configuring the URLConnection-based HTTP client

The AWS SDK for Java 2.x offers a lighter-weight `UrlConnectionHttpClient` HTTP client in comparison to the default `ApacheHttpClient`. The `UrlConnectionHttpClient` is based on Java's `URLConnection`.

The `UrlConnectionHttpClient` loads more quickly than the Apache-based HTTP client, but has fewer features. Because it loads more quickly, it is a good solution (p. 47) for Java AWS Lambda functions.

The `UrlConnectionHttpClient` has several configurable options that you can access.

To learn how to configure the Apache-based HTTP client, see Configuring the Apache-based HTTP client (p. 52).

> **Note**
> The `UrlConnectionHttpClient` does not support the HTTP PATCH method.
> A handful of AWS API operations require PATCH requests. Those operation names usually start with `Update*`. The following are several examples.
>
> - Several `Update*` operations in the AWS Security Hub API and also the BatchUpdateFindings operation
> - All Amazon API Gateway API `Update*` operations
> - Several `Update*` operations in the Amazon WorkDocs API
>
> If you might use the `UrlConnectionHttpClient`, first refer to the API Reference for the AWS service that you're using. Check to see if the operations you need use the PATCH operation.

Accessing the `UrlConnectionHttpClient`

To configure and use the `UrlConnectionHttpClient`, you declare a dependency on the `url-connection-client` Maven artifact in your `pom.xml` file.

Unlike the `ApacheHttpClient`, the `UrlConnectionHttpClient` is not automatically added to your project, so use must specifically declare it.

The following example of a `pom.xml` file shows the dependencies required to use and configure the HTTP client.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.17.290</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<!-- other dependencies such as s3 or dynamodb -->

<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>url-connection-client</artifactId>
    </dependency>
</dependencies>
```

Configuring the `UrlConnectionHttpClient`

You can configure an instance of `UrlConnectionHttpClient` along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use the UrlConnectionHttpClient.Builder to configure the properties for the URLConnection-based HTTP client.

Best practice: dedicate an `UrlConnectionHttpClient` instance to a service client

If you need to configure an instance of the `UrlConnectionHttpClient`, we recommend that you build the dedicated `UrlConnectionHttpClient` instance. You can do so by using the `httpClientBuilder` method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the `UrlConnectionHttpClient` instance is not closed down when it's no longer needed.

The following example creates an `S3Client` and configures the embedded instance of `UrlConnectionHttpClient` with `maxConnections` and `connectionTimeout` values. The HTTP instance is created using the `httpClientBuilder` method of `S3Client.Builder`.

The following example creates an `S3Client` and configures the embedded instance of `UrlConnectionHttpClient` with `socketTimeout` and `proxyConfiguration` values. The `proxyConfiguration` method takes a Java lambda expression of type `Consumer<ProxyConfiguration.Builder>`. The HTTP instance is created using the `httpClientBuilder` method of `S3Client.Builder`.

Imports

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient;
import java.net.URI;
import java.time.Duration;
```

Code

```
// Singleton: Use the s3Client for all requests.
S3Client s3Client =
    S3Client.builder()
            .httpClientBuilder(UrlConnectionHttpClient.builder()
                    .socketTimeout(Duration.ofMinutes(5))
                    .proxyConfiguration(proxy -> proxy.endpoint(URI.create("http://
proxy.mydomain.net:8888"))))
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

// Perform work with the s3Client.

s3Client.close();   // Requests completed: Close the s3client.
```

Alternative approach: share an `UrlConnectionHttpClient` instance

To help keep resource and memory usage lower for your application, you can configure an `UrlConnectionHttpClient` and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
> When an `UrlConnectionHttpClient` instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures an URLConnection-based HTTP client, which is used by two service clients. The configured `UrlConnectionHttpClient` instance is passed to the `httpClient` method of the service client's builder. When the service clients and the HTTP client are no longer needed, they are explicitly closed. The HTTP client is closed last.

Imports

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.urlconnection.ProxyConfiguration;
import software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.net.URI;
import java.time.Duration;
```

Code

```
SdkHttpClient urlHttpClient =
    UrlConnectionHttpClient.builder()
        .socketTimeout(Duration.ofMinutes(5))
        .proxyConfiguration((ProxyConfiguration.Builder proxy) ->
 proxy.endpoint(URI.create("http://proxy.mydomain.net:8888")))
        .build();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
```

```
S3Client s3Client =
    S3Client.builder()
            .httpClient(urlHttpClient)
            .defaultsMode(DefaultsMode.IN_REGION)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
                   .httpClient(urlHttpClient)
                   .defaultsMode(DefaultsMode.IN_REGION)
                   .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                   .build();

// Perform work with the s3Client and dynamoDbClient.

// Requests completed: Close all service clients.
s3Client.close();
dynamoDbClient.close();
urlHttpClient.close();
```

Using `URLConnectionHttpClient` and `ApacheHttpClient` together

When using the `UrlConnectionHttpClient` in your application, you must supply each service client with either a `URLConnectionHttpClient` instance or a `ApacheHttpClient` instance using the service client builder's `httpClientBuilder` method.

An exception occurs if your program uses multiple service clients and both of the following are true:

- One service client is configured to use a `UrlConnectionHttpClient` instance
- Another service client uses the default `ApacheHttpClient` without explicitly building it using `httpClientConfig`

The exception will state that multiple HTTP implementations were found on the classpath.

The following example code snippet leads to an exception.

```
// The dynamoDbClient uses the UrlConnectionHttpClient
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
        .httpClient(UrlConnectionHttpClient.create())
        .build();

// The s3Client below uses the ApacheHttpClient at runtime, without specifying it.
// An SdkClientException is thrown with the message that multiple HTTP implementations were
 found on the classpath.
S3Client s3Client = S3Client.builder().build();

// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

Avoid the exception by explicitly configuring the `S3Client` with an `ApacheHttpClient`.

```
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
        .httpClient(UrlConnectionHttpClient.create())
        .build();

S3Client s3Client = S3Client.builder()
        .httpClient(ApacheHttpClient.create())    // Explicitly build the ApacheHttpClient.
        .build();
```

```
// Perform work with the s3Client and dynamoDbClient.

dynamoDbClient.close();
s3Client.close();
```

Note
To explicitly create the `ApacheHttpClient`, you must add a dependency (p. 52) on the `apache-client` artifact in your Maven project file.

Configuring the Netty-based HTTP client

The default HTTP client for asynchronous operations in the AWS SDK for Java 2.x is the Netty-based NettyNioAsyncHttpClient. The Netty-based client is based on the asynchronous event-driven network framework of the Netty project.

As an alternative, you can use the new AWS CRT-based HTTP client (p. 62). This topic shows you how to configure the `NettyNioAsyncHttpClient`.

Accessing the `NettyNioAsyncHttpClient`

In most situations, you use the `NettyNioAsyncHttpClient` without any explicit configuration in asynchronous programs. You declare your asynchronous service clients and the SDK will configure the `NettyNioAsyncHttpClient` with standard values for you.

If you want to explicitly configure the `NettyNioAsyncHttpClient` or use it with multiple service clients, you need to make it available for configuration.

No configuration needed

When you declare a dependency on a service client in Maven, the SDK adds a *runtime* dependency on the `netty-nio-client` artifact. This makes the `NettyNioAsyncHttpClient` class available to your code at runtime. If you are not configuring the Netty-based HTTP client, you don't need to specify a dependency for it.

In the following XML snippet of a Maven `pom.xml` file, the dependency declared with `<artifactId>dynamodb-enhanced</artifactId>` transitively brings in the Netty-based HTTP client. You don't need to declare a dependency specifically for it.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.17.290</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
</dependencies>
```

With these dependencies, you cannot make any HTTP configuration changes, since the `NettyNioAsyncHttpClient` library is only on the runtime classpath.

Configuration needed

To configure the `NettyNioAsyncHttpClient`, you need to add a dependency on the `netty-nio-client` artifact at *compile* time.

Refer to the following example of a Maven `pom.xml` file to configure the `NettyNioAsyncHttpClient`.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.17.290</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>dynamodb-enhanced</artifactId>
        </dependency>
        <!-- By adding the netty-nio-client dependency, NettyNioAsyncHttpClient will be
             added to the compile classpath so you can configure it. -->
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>netty-nio-client</artifactId>
        </dependency>
    </dependencies>
```

Configuring the `NettyNioAsyncHttpClient`

You can configure an instance of `NettyNioAsyncHttpClient` along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use the NettyNioAsyncHttpClient.Builder to configure the properties for the Netty-based HTTP client instance.

Best practice: dedicate a `NettyNioAsyncHttpClient` instance to a service client

If you need to configure an instance of the `NettyNioAsyncHttpClient`, we recommend that you build the dedicated `NettyNioAsyncHttpClient` instance. You can do so by using the `httpClientBuilder` method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the `NettyNioAsyncHttpClient` instance is not closed down when it's no longer needed.

The following example creates a `DynamoDbAsyncClient` instance, which is also used by a `DynamoDbEnhancedAsyncClient` instance. The `DynamoDbAsyncClient` instance contains the `NettyNioAsyncHttpClient` instance with `connectionTimeout` and `maxConcurrency` values. The HTTP instance is created using `httpClientBuilder` method of `DynamoDbAsyncClient.Builder`.

Imports

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedAsyncClient;
import
 software.amazon.awssdk.enhanced.dynamodb.extensions.AutoGeneratedTimestampRecordExtension;
```

```
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.time.Duration;
```

Code

```
// DynamoDbAsyncClient is the lower-level client used by the enhanced client.
DynamoDbAsyncClient dynamoDbAsyncClient =
    DynamoDbAsyncClient
        .builder()
        .httpClientBuilder(NettyNioAsyncHttpClient.builder()
            .connectionTimeout(Duration.ofMinutes(5))
            .maxConcurrency(100))
        .defaultsMode(DefaultsMode.IN_REGION)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

// Singleton: Use dynamoDbAsyncClient and enhancedClient for all requests.
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient
        .builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .extensions(AutoGeneratedTimestampRecordExtension.create())
        .build();

// Perform work with the dynamoDbAsyncClient and enhancedClient.

// Requests completed: Close dynamoDbAsyncClient.
dynamoDbAsyncClient.close();
```

Alternative approach: share a `NettyNioAsyncHttpClient` instance

To help keep resource and memory usage lower for your application, you can configure a `NettyNioAsyncHttpClient` and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
> When a `NettyNioAsyncHttpClient` instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures a Netty-based HTTP client, which is used by two service clients. The configured `NettyNioAsyncHttpClient` instance is passed to the `httpClient` method of each service client's builder. When the service clients and the HTTP client are no longer needed, they are explicitly closed. The HTTP client is closed last.

Imports

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

Code

```
// Create a NettyNioAsyncHttpClient shared instance.
SdkAsyncHttpClient nettyHttpClient =
    NettyNioAsyncHttpClient.builder()
                           .connectionTimeout(Duration.ofMinutes(5))
                           .maxConcurrency(100)
                           .build();
```

```
// Singletons: Use the s3AsyncClient, dbAsyncClient, and enhancedAsyncClient for all
 requests.
S3AsyncClient s3AsyncClient =
    S3AsyncClient.builder()
                 .httpClient(nettyHttpClient)
                 .build();

DynamoDbAsyncClient dbAsyncClient =
    DynamoDbAsyncClient.builder()
                       .httpClient(nettyHttpClient)
                       .defaultsMode(DefaultsMode.IN_REGION)

 .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                       .build();

DynamoDbEnhancedAsyncClient enhancedAsyncClient =
    DynamoDbEnhancedAsyncClient.builder()
                               .dynamoDbClient(dbAsyncClient)
                               .extensions(AutoGeneratedTimestampRecordExtension.create())
                               .build();

// Perform work with s3AsyncClient, dbAsyncClient, and enhancedAsyncClient.

// Requests completed: Close all service clients.
s3AsyncClient.close();
dbAsyncClient.close()
nettyHttpClient.close();  // Explicitly close nettyHttpClient.
```

Configuring the AWS CRT-based HTTP client

The `AwsCrtAsyncHttpClient` is a new asynchronous HTTP client that you can use with the AWS SDK for Java 2.x. The `AwsCrtAsyncHttpClient` brings the following benefits for an HTTP client to the SDK.

- faster SDK startup time
- smaller memory footprint
- reduced latency time
- connection health management
- DNS load balancing

This topic is about using and configuring the AWS CRT-based HTTP client. For information about configuring the Netty-based HTTP client, see the previous topic (p. 59).

AWS CRT-based components in the SDK

The AWS CRT-based *HTTP* client, described in this topic, and the AWS CRT-based *S3* client are different components in the SDK.

The **AWS CRT-based HTTP client** is an implementation of the SdkAsyncHttpClient interface and is used for general HTTP communication. It is an alternative to the Netty implementation of the `SdkAsyncHttpClient` interface and offers several benefits.

The **AWS CRT-based S3 client (p. 94)** is an implementation of the S3AsyncClient interface and is used for working with the Amazon S3 service. It is an alternative to the Java-based implementation of the `S3AsyncClient` interface and offers several advantages.

Although both components use libraries from the AWS Common Runtime, the AWS CRT-based HTTP client does not use the aws-c-s3 library and does not support the S3 multipart upload API features. The AWS CRT-based S3 client, by contrast, was purpose-built to support the S3 multipart upload API features.

Accessing the `AwsCrtAsyncHttpClient`

Before you can use the AWS CRT-based HTTP client, add the `aws-crt-client` artifact to your project's dependencies.

The following Maven `pom.xml` shows the AWS CRT-based HTTP client declared using the bill of materials (BOM) mechanism.

```
<project>
   <properties>
     <aws.sdk.version>VERSION</aws.sdk.version>
  </properties>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
   </dependency>
  </dependencies>
</project>
```

Visit the Maven central repository for the latest *VERSION* value.

Configuring the `AwsCrtAsyncHttpClient`

You can configure an instance of `AwsCrtAsyncHttpClient` along with building a service client, or you can configure a single instance to share across multiple service clients.

With either approach, you use the AwsCrtAsyncHttpClient.Builder to configure the properties for the AWS CRT-based HTTP client instance.

Best practice: dedicate a `AwsCrtAsyncHttpClient` instance to a service client

If you need to configure an instance of the `AwsCrtAsyncHttpClient`, we recommend that you build the dedicated `AwsCrtAsyncHttpClient` instance. You can do so by using the `httpClientBuilder` method of the service client's builder. This way, the lifecycle of the HTTP client is managed by the SDK, which helps avoid potential memory leaks if the `AwsCrtAsyncHttpClient` instance is not closed down when it's no longer needed.

The following example creates an `S3Client` and configures the `AwsCrtAsyncHttpClient` with `connectionTimeout` and `maxConcurrency` values.

Imports

```
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

Code

```
// Singleton: Use s3AsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClientBuilder(AwsCrtAsyncHttpClient
            .builder()
            .connectionTimeout(Duration.ofSeconds(3))
            .maxConcurrency(100))
    .build();

// Perform work with the s3AsyncClient.

// Requests completed: Close the s3AsyncClient.
s3AsyncClient.close();
```

Alternative approach: share a `AwsCrtAsyncHttpClient` instance

To help keep resource and memory usage lower for your application, you can configure a `AwsCrtAsyncHttpClient` and share it across multiple service clients. The HTTP connection pool will be shared, which lowers resource usage.

> **Note**
> When a `AwsCrtAsyncHttpClient` instance is shared, you must close it when it is ready to be disposed. The SDK will not close the instance when the service client is closed.

The following example configures a AWS CRT-based HTTP client instance with `connectionTimeout` and `maxConcurrency` values. The configured `AwsCrtAsyncHttpClient` instance is passed to the `httpClient` method of each service client's builder. When the service clients and the HTTP client are no longer needed, they are explicitly closed. The HTTP client is closed last.

Imports

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.awscore.defaultsmode.DefaultsMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import java.time.Duration;
```

Code

```
// Create an AwsCrtAsyncHttpClient shared instance.
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(3))
    .maxConcurrency(100)
    .build();

// Singletons: Use the s3AsyncClient and dynamoDbAsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClient(crtAsyncHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();

DynamoDbAsyncClient dynamoDbAsyncClient = DynamoDbAsyncClient.builder()
    .httpClient(crtAsyncHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();
```

```
// Requests completed: Close all service clients.
s3AsyncClient.close();
dynamoDbAsyncClient.close()
crtAsyncHttpClient.close();  // Explicitly close crtAsyncHttpClient.
```

Setting the AWS CRT-based HTTP client as the default

For asynchronous operations in the AWS SDK for Java 2.x, you can replace the `NettyNioAsyncHttpClient` as the default asynchronous HTTP client in your program with the `AwsCrtAsyncHttpClient`.

You set this in your project's Maven `pom.xml` file by excluding the dependency on the `netty-nio-client` for each service client. Alternatively, you can set the default HTTP client by using a Java system property when you run your app or in your application code.

Remove Netty from the project dependencies

The following `pom.xml` example removes the Netty-based HTTP client from the classpath so that the AWS CRT-based HTTP client will be used instead.

```
<project>
   <properties>
     <aws.sdk.version>VERSION</aws.sdk.version>
  </properties>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3</artifactId>
      <version>${aws.sdk.version}</version>
      <exclusions>
         <exclusion>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>netty-nio-client</artifactId>
         </exclusion>
      </exclusions>
   </dependency>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
   </dependency>
  </dependencies>
</project>
```

Visit the Maven central repository for the latest *VERSION* value.

> **Note**
> If multiple service clients are declared in a `pom.xml` file, all require the `exclusions`XML element.

Setting via Java system property

To use the AWS CRT-based HTTP client as the default HTTP for your application, you can set the Java system property `software.amazon.awssdk.http.async.service.impl` to a value of `software.amazon.awssdk.http.crt.AwsCrtSdkHttpService`.

To set during application startup, run a command similar to the following.

```
java app.jar -Dsoftware.amazon.awssdk.http.async.service.impl=\
```

```
software.amazon.awssdk.http.crt.AwsCrtSdkHttpService
```

Use the following code snippet to set the system property in your application code.

```
System.setProperty("software.amazon.awssdk.http.async.service.impl",
"software.amazon.awssdk.http.crt.AwsCrtSdkHttpService");
```

Note
You need to add a dependency on the `aws-crt-client` artifact in your `pom1.xml` file when you use a system property to configure the use of the AWS CRT-based HTTP client.

Advanced configuration of `AwsCrtAsyncHttpClient`

You can use the AWS CRT-based HTTP client to configure various settings, including connection health configuration and maximum idle time. You can review the configuration options available for the `AwsCrtAsyncHttpClient`.

Connection health configuration

You can configure connection health configuration for the AWS CRT-based HTTP client using the `connectionHealthConfiguration` method on the HTTP client builder.

The following example creates an `S3AsyncClient` that uses a `AwsCrtAsyncHttpClient` instance configured with connection health configuration and a maximum idle time for connections.

```
// Singleton: Use the s3AsyncClient for all requests.
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .httpClientBuilder(AwsCrtAsyncHttpClient
        .builder()
        .connectionHealthConfiguration(builder -> builder
              .minimumThroughputInBps(32000L)
              .minimumThroughputTimeout(Duration.ofSeconds(3)))
         .connectionMaxIdleTime(Duration.ofSeconds(5)))
    .build();

// Perform work with s3AsyncClient.

// Requests complete: Close the service client.
s3AsyncClient.close();
```

HTTP/2 support

The HTTP/2 protocol is not yet supported in the AWS CRT-based HTTP client, but is planned for a future release.

In the meantime, if you are using service clients that require HTTP/2 support such as the KinesisAsyncClient or the TranscribeStreamingAsyncClient, consider using the NettyNioAsyncHttpClient instead.

Exception handling for the AWS SDK for Java 2.x

Understanding how and when the AWS SDK for Java 2.x throws exceptions is important to building high-quality applications using the SDK. The following sections describe the different cases of exceptions that are thrown by the SDK and how to handle them appropriately.

Why unchecked exceptions?

The AWS SDK for Java uses runtime (or unchecked) exceptions instead of checked exceptions for these reasons:

- To allow developers fine-grained control over the errors they want to handle without forcing them to handle exceptional cases they aren't concerned about (and making their code overly verbose)
- To prevent scalability issues inherent with checked exceptions in large applications

In general, checked exceptions work well on small scales, but can become troublesome as applications grow and become more complex.

AwsServiceException (and subclasses)

AwsServiceException is the most common exception that you'll experience when using the AWS SDK for Java. `AwsServiceException` is a subclass of the more general SdkServiceException. `AwsServiceExceptions` represent an error response from an AWS service. For example, if you try to terminate an Amazon EC2 instance that doesn't exist, Amazon EC2 will return an error response and all the details of that error response will be included in the `AwsServiceException` that's thrown.

When you encounter an `AwsServiceException`, you know that your request was successfully sent to the AWS service but couldn't be successfully processed. This can be because of errors in the request's parameters or because of issues on the service side.

`AwsServiceException` provides you with information such as:

- Returned HTTP status code
- Returned AWS error code
- Detailed error message from the service in the AwsErrorDetails class
- AWS request ID for the failed request

In most cases, a service-specific subclass of `AwsServiceException` is thrown to allow developers fine-grained control over handling error cases through catch blocks. The Java SDK API reference for AwsServiceException displays the large number of `AwsServiceException` subclasses. Use the subclass links to drill down to see the granular exceptions thrown by a service.

For example, the following links to the SDK API reference show the exception hierarchies for a few common AWS services. The list of subclasses shown on each pages shows the specific exceptions that your code can catch.

- Amazon S3
- DynamoDB
- Amazon SQS

To learn more about an exception, inspect the `errorCode` on the AwsErrorDetails object. You can use the `errorCode` value to look up information in the service guide API. For example if an `S3Exception` is caught and the `AwsErrorDetails#errorCode()` value is `InvalidRequest`, use the list of error codes in the Amazon S3 API Reference to see more details.

SdkClientException

SdkClientException indicates that a problem occurred inside the Java client code, either while trying to send a request to AWS or while trying to parse a response from AWS. An `SdkClientException`

is generally more severe than an `SdkServiceException`, and indicates a major problem that is preventing the client from making service calls to AWS services. For example, the AWS SDK for Java throws an `SdkClientException` if no network connection is available when you try to call an operation on one of the clients.

Logging with the SDK for Java 2.x

The AWS SDK for Java 2.x uses SLF4J, which is an abstraction layer that enables the use of any one of several logging systems at runtime.

Supported logging systems include the Java Logging Framework and Apache Log4j 2, among others. This topic shows you how to use Log4j 2 as the logging system for working with the SDK.

Log4j 2 configuration file

You typically use a configuration file, named`log4j2.xml` with Log4j 2. Example configuration files are shown below. To learn more about the values used in the configuration file, see the manual for Log4j configuration.

Place the `log4j2.xml` file in the `<project-dir>/src/main/resources` directory when using Maven.

The `log4j2.xml` configuration file specifies properties such as logging level, where logging output is sent (for example, to a file or to the console), and the format of the output. The logging level specifies the level of detail that Log4j 2 outputs. Log4j 2 supports the concept of multiple logging *hierarchies*. The logging level is set independently for each hierarchy. The main logging hierarchy that you use with the AWS SDK for Java 2.x is `software.amazon.awssdk`.

Setting dependencies

To configure the Log4j 2 binding for SLF4J in Maven, use the following in your `pom.xml` file:

```
...
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-bom</artifactId>
            <version>VERSION</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
     </dependencies>
</dependencyManagement>
...
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-core</artifactId>
</dependency>
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-api</artifactId>
</dependency>
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-slf4j2-impl</artifactId>
</dependency>
```

```
...
```

Replace `VERSION` above with the latest version from Maven central.

SDK-specific errors and warnings

We recommend that you always leave the "software.amazon.awssdk" logger hierarchy set to "WARN" to catch any important messages from the SDK's client libraries. For example, if the Amazon S3 client detects that your application hasn't properly closed an `InputStream` and could be leaking resources, the S3 client reports it through a warning message to the logs. This also ensures that messages are logged if the client has any problems handling requests or responses.

The following `log4j2.xml` file sets the `rootLogger` to "WARN", which causes warning and error-level messages from all loggers in the application to be output, *including* those in the "software.amazon.awssdk" hierarchy. Alternatively, you can explicitly set the "software.amazon.awssdk" logger hierarchy to "WARN" if `<Root level="ERROR">` is used.

Example Log4j2.xml configuration file

This configuration will log messages at the "ERROR" and "WARN" levels to the console for all logger hierarchies.

```
<Configuration status="WARN">
 <Appenders>
  <Console name="ConsoleAppender" target="SYSTEM_OUT">
   <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
  </Console>
 </Appenders>

 <Loggers>
  <Root level="WARN">
   <AppenderRef ref="ConsoleAppender"/>
  </Root>
 </Loggers>
</Configuration>
```

Request/response summary logging

Every request to an AWS service generates a unique AWS request ID that is useful if you run into an issue with how an AWS service is handling a request. AWS request IDs are accessible programmatically through `SdkServiceException` objects in the SDK for any failed service call, and can also be reported through the "DEBUG" log level of the "software.amazon.awssdk.request" logger.

The following `log4j2.xml` file enables a summary of requests and responses.

```
<Configuration status="WARN">
 <Appenders>
  <Console name="ConsoleAppender" target="SYSTEM_OUT">
   <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
  </Console>
 </Appenders>

 <Loggers>
  <Root level="ERROR">
   <AppenderRef ref="ConsoleAppender"/>
  </Root>
  <Logger name="software.amazon.awssdk" level="WARN" />
  <Logger name="software.amazon.awssdk.request" level="DEBUG" />
 </Loggers>
```

```
</Configuration>
```

Here is an example of the log output:

```
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
 DefaultSdkHttpFullRequest(httpMethod=POST, protocol=https, host=dynamodb.us-
east-1.amazonaws.com, encodedPath=/, headers=[amz-sdk-invocation-id, Content-Length,
 Content-Type, User-Agent, X-Amz-Target], queryParameters=[])
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Received successful
 response: 200, Request ID: QS9DUMME2NHEDH8TGT9N5V53OJVV4KQNSO5AEMVJF66Q9ASUAAJG, Extended
 Request ID: not available
```

If you are interested in only the request ID use `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

Verbose wire logging

It can be useful to see the exact requests and responses that the SDK for Java 2.x sends and receives. If you need access to this information, you can temporarily enable it by adding the necessary configuration depending on the HTTP client the service client uses.

By default, synchronous service clients, such as the S3Client, use an underlying Apache HttpClient, and asynchronous service clients, such as the S3AsyncClient, use a Netty non-blocking HTTP client.

Here is a breakdown of HTTP clients you can use for the two categories of service clients:

Synchronous HTTP Clients	Asynchronous HTTP Clients
ApacheHttpClient (default)	NettyNioAsyncHttpClient (default)
UrlConnectionHttpClient	AwsCrtAsyncHttpClient

Consult the appropriate tab below for configuration settings you need to add depending on the underlying HTTP client.

> **Warning**
> We recommend you only use wire logging for debugging purposes. Disable it in your production environments because it can log sensitive data. It logs the full request or response without encryption, even for an HTTPS call. For large requests (e.g., to upload a file to Amazon S3) or responses, verbose wire logging can also significantly impact your application's performance.

ApacheHttpClient

Add the "org.apache.http.wire" logger to the `log4j2.xml` configuration file and set the level to "DEBUG".

The following `log4j2.xml` file turns on full wire logging for the Apache HttpClient.

```
<Configuration status="WARN">
 <Appenders>
  <Console name="ConsoleAppender" target="SYSTEM_OUT">
   <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
  </Console>
 </Appenders>

 <Loggers>
  <Root level="WARN">
```

```
   <AppenderRef ref="ConsoleAppender"/>
  </Root>
  <Logger name="software.amazon.awssdk" level="WARN" />
  <Logger name="software.amazon.awssdk.request" level="DEBUG" />
  <Logger name="org.apache.http.wire" level="DEBUG" />
 </Loggers>
</Configuration>
```

An additional Maven dependency on the `log4j-1.2-api` artifact is required for wire logging with Apache as it uses 1.2 under the hood. Add the following to the `pom.xml` file to enable wire logging.

```
<dependency>
 <groupId>org.apache.logging.log4j</groupId>
 <artifactId>log4j-1.2-api</artifactId>
</dependency>
```

The full set of Maven dependencies for using log4j 2, including wire logging for the Apache HttpClient, is:

```
...
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-bom</artifactId>
            <version>VERSION</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
     </dependencies>
</dependencyManagement>
...
<!-- The following 3 entries are needed for Log4j2 with SLF4J -->
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-core</artifactId>
</dependency>
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-api</artifactId>
</dependency>
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-slf4j2-impl</artifactId>
</dependency>

<!-- The following is needed for Apache HttpClient wire logging -->
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-1.2-api</artifactId>
</dependency>
...
```

UrlConnectionHttpClient

To log details for service clients that use the `UrlConnectionHttpClient`, first create a `logging.properties` file with the following contents:

```
handlers=java.util.logging.ConsoleHandler
java.util.logging.ConsoleHandler.level=FINEST
sun.net.www.protocol.http.HttpURLConnection.level=ALL
```

Set the following JVM system property with the full path of the `logging.properties`:

```
-Djava.util.logging.config.file=/full/path/to/logging.properties
```

This configuration will log the only the headers of the request and response, for example:

```
<Request>  FINE: sun.net.www.MessageHeader@35a9782c11 pairs: {GET /fileuploadtest
 HTTP/1.1: null}{amz-sdk-invocation-id: 5f7e707e-4ac5-bef5-ba62-00d71034ffdc}
{amz-sdk-request: attempt=1; max=4}{Authorization: AWS4-HMAC-SHA256
 Credential=<deleted>/20220927/us-east-1/s3/aws4_request, SignedHeaders=amz-sdk-
invocation-id;amz-sdk-request;host;x-amz-content-sha256;x-amz-date;x-amz-te,
 Signature=e367fa0bc217a6a65675bb743e1280cf12fbe8d566196a816d948fdf0b42ca1a}{User-
Agent: aws-sdk-java/2.17.230 Mac_OS_X/12.5 OpenJDK_64-Bit_Server_VM/25.332-b08
 Java/1.8.0_332 vendor/Amazon.com_Inc. io/sync http/UrlConnection cfg/retry-mode/
legacy}{x-amz-content-sha256: UNSIGNED-PAYLOAD}{X-Amz-Date: 20220927T133955Z}{x-amz-te:
 append-md5}{Host: tkhill-test1.s3.amazonaws.com}{Accept: text/html, image/gif, image/
jpeg, *; q=.2, */*; q=.2}{Connection: keep-alive}
<Response> FINE: sun.net.www.MessageHeader@70a36a6611 pairs: {null: HTTP/1.1 200 OK}{x-
amz-id-2: sAFeZDOKdUMsBbkdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/dFuOvr2tUb7Y1wEHGdJ3DSIxq0=}
{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue, 27 Sep 2022 13:39:57 GMT}{Last-
Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag: "2cbe5ad4a064cedec33b452bebf48032"}
{x-amz-transfer-encoding: append-md5}{Accept-Ranges: bytes}{Content-Type: text/plain}
{Server: AmazonS3}{Content-Length: 67}
```

To see the request/response bodies, add `-Djavax.net.debug=all` to the JVM properties. This additional property logs a great deal of information, including all SSL information.

Within the log console or log file, search for `"GET"` or `"POST"` to quickly go to the section of the log containing actual requests and responses. Search for `"Plaintext before ENCRYPTION"` for requests and `"Plaintext after DECRYPTION"` for responses to see the full text of the headers and bodies.

NettyNioAsyncHttpClient

If your asynchronous service client uses the default `NettyNioAsyncHttpClient`, add two additional loggers to your `log4j2.xml` file to log HTTP headers and request/response bodies.

```
<Logger name="io.netty.handler.logging" level="DEBUG" />
<Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" />
```

Here is a complete `log4j2.xml` example:

```
<Configuration status="WARN">
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{YYYY-MM-dd HH:mm:ss} [%t] %-5p %c:%L - %m%n" />
        </Console>
    </Appenders>

    <Loggers>
        <Root level="WARN">
            <AppenderRef ref="ConsoleAppender"/>
        </Root>
        <Logger name="software.amazon.awssdk" level="WARN" />
        <Logger name="software.amazon.awssdk.request" level="DEBUG" />
        <Logger name="io.netty.handler.logging" level="DEBUG" />
        <Logger name="io.netty.handler.codec.http2.Http2FrameLogger" level="DEBUG" />
    </Loggers>
</Configuration>
```

These settings log all header details and request/response bodies.

AwsCrtAsyncHttpClient

If you have configured your service client to use an instance of `AwsCrtAsyncHttpClient`, you can log details by setting JVM system properties or programmatically.

Log to a file at "Debug" level	
Using system properties:	Programmatically:

`-Daws.crt.log.level=Trace -Daws.crt.log.destination=File -Daws.crt.log.filename=<path to file>`	`import import software.amazon.awssdk.crt.Log;`
	
	`// Execute this statement before constructing the SDK service client.`
	`Log.initLoggingToFile(Log.LogLevel.Trace, "<path to file>");`

Log to the console at "Debug" level	
Using system properties:	Programmatically:

`-Daws.crt.log.level=Trace -Daws.crt.log.destination=Stdout`	`import import software.amazon.awssdk.crt.Log;`
	
	`// Execute this statement before constructing the SDK service client.`
	`Log.initLoggingToStdout(Log.LogLevel.Trace);`

For security reasons, at the "Trace" level the `AwsCrtAsyncHttpClient` logs only response headers. Request headers, request bodies, and response bodies are not logged.

Setting the JVM TTL for DNS name lookups

The Java virtual machine (JVM) caches DNS name lookups. When the JVM resolves a hostname to an IP address, it caches the IP address for a specified period of time, known as the *time-to-live* (TTL).

Because AWS resources use DNS name entries that occasionally change, we recommend that you configure your JVM with a TTL value of no more than 60 seconds. This ensures that when a resource's IP address changes, your application will be able to receive and use the resource's new IP address by requerying the DNS.

On some Java configurations, the JVM default TTL is set so that it will *never* refresh DNS entries until the JVM is restarted. Thus, if the IP address for an AWS resource changes while your application is still running, it won't be able to use that resource until you *manually restart* the JVM and the cached IP information is refreshed. In this case, it's crucial to set the JVM's TTL so that it will periodically refresh its cached IP information.

Note
The default TTL can vary according to the version of your JVM and whether a security manager is installed. Many JVMs provide a default TTL less than 60 seconds. If you're using such a JVM and not using a security manager, you can ignore the remainder of this topic.

How to set the JVM TTL

To modify the JVM's TTL, set the networkaddress.cache.ttl property value. Use one of the following methods, depending on your needs:

- **globally, for all applications that use the JVM**. Set `networkaddress.cache.ttl` in the `$JAVA_HOME/jre/lib/security/java.security` file:

```
networkaddress.cache.ttl=60
```

- **for your application only**, set `networkaddress.cache.ttl` in your application's initialization code:

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "60");
```

Best practices for AWS SDK for Java 2.x

This section lists best practices for using the SDK for Java 2.x.

Topics

- Reuse an SDK client, if possible (p. 74)
- Close input streams from client operations (p. 74)
- Tune HTTP configurations based on performance tests (p. 74)
- Use OpenSSL for the Netty-based HTTP client (p. 75)
- Configure API timeouts (p. 75)
- Use metrics (p. 76)

Reuse an SDK client, if possible

Each SDK client maintains its own HTTP connection pool. A connection that already exists in the pool can be reused by a new request to cut down the time to establish a new connection. We recommend sharing a single instance of the client to avoid the overhead of having too many connection pools that aren't used effectively. All SDK clients are thread safe.

If you don't want to share a client instance, call `close()` on the instance to release the resources when the client is not needed.

Close input streams from client operations

For streaming operations such as `S3Client#getObject`, if you are working with `ResponseInputStream` directly, we recommend that you do the following:

- Read all the data from the input stream as soon as possible.
- Close the input stream as soon as possible.

We make these recommendations because the input stream is a direct stream of data from the HTTP connection and the underlying HTTP connection can't be reused until all data from the stream has been read and the stream is closed. If these rules are not followed, the client can run out of resources by allocating too many open, but unused, HTTP connections.

Tune HTTP configurations based on performance tests

The SDK provides a set of default http configurations that apply to general use cases. We recommend that customers tune HTTP configurations for their applications based on their use cases.

As a good starting point, the SDK offers a smart configuration defaults (p. 51) feature. This feature is available starting with version 2.17.102. You choose a mode depending on your use case, which provides sensible configuration values.

Use OpenSSL for the Netty-based HTTP client

By default, the SDK's `NettyNioAsyncHttpClient` uses the JDK's default SSL implementation as the `SslProvider`. Our testing found that OpenSSL performs better than JDK's default implementation. The Netty community also recommends using OpenSSL.

To use OpenSSL, add `netty-tcnative` to your dependencies. For configuration details, see the Netty project documentation.

After you have `netty-tcnative` configured for your project, the `NettyNioAsyncHttpClient` instance will automatically select OpenSSL. Alternatively, you can set the `SslProvider` explicitly using the `NettyNioAsyncHttpClient` builder as shown in the following snippet.

```
NettyNioAsyncHttpClient.builder()
                        .sslProvider(SslProvider.OPENSSL)
                        .build();
```

Configure API timeouts

The SDK provides default values for some timeout options, such as connection timeout and socket timeouts, but not for API call timeouts or individual API call attempt timeouts. It is a good practice to set timeouts for both the individual attempts and the entire request. This will ensure your application fails fast in an optimal way when there are transient issues that could cause request attempts to take longer to complete or fatal network issues.

You can configure timeouts for all requests made by a service clients using `ClientOverrideConfiguration#apiCallAttemptTimeout` and `ClientOverrideConfiguration#apiCallTimeout`.

The following example shows the configuration of an Amazon S3 client with custom timeout values.

```
S3Client.builder()
        .overrideConfiguration(
             b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
                   .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))
        .build();
```

`apiCallAttemptTimeout`

This setting sets the amount of time for a single HTTP attempt, after which the API call can be retried.

`apiCallTimeout`

The value for this property configures the amount of time for the entire execution, including all retry attempts.

As an alternative to setting these timeout values on the service client, you can use `RequestOverrideConfiguration#apiCallTimeout()` and `RequestOverrideConfiguration#apiCallAttemptTimeout()` to configure a single request .

The following example configures a single `listBuckets` request with custom timeout values.

```
s3Client.listBuckets(lbr -> lbr.overrideConfiguration(
        b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
               .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))));
```

When you use these properties together, you set a hard limit on the total time spent on all attempts across retries. You also set an individual HTTP request to fail fast on a slow request.

Use metrics

The SDK for Java can collect metrics (p. 83) for the service clients in your application. You can use these metrics to identify performance issues, review overall usage trends, review service client exceptions returned, or to dig in to understand a particular issue.

We recommend that you collect metrics, then analyze the Amazon CloudWatch Logs, in order to gain a deeper understanding of your application's performance.

Features of the AWS SDK for Java 2.x

This section provides information about the features of the AWS SDK for Java 2.x.

Topics

- Asynchronous programming (p. 77)
- Using the DynamoDB Enhanced Client in the AWS SDK for Java 2.x (p. 83)
- Working with HTTP/2 in the AWS SDK for Java (p. 83)
- Enabling SDK metrics for the AWS SDK for Java (p. 83)
- Retrieving paginated results using the AWS SDK for Java 2.x (p. 88)
- AWS CRT-based S3 client (p. 94)
- Amazon S3 Transfer Manager (p. 95)
- Using waiters in the AWS SDK for Java 2.x (p. 101)

Asynchronous programming

The AWS SDK for Java 2.x features truly nonblocking asynchronous clients that implement high concurrency across a few threads. The AWS SDK for Java 1.x has asynchronous clients that are wrappers around a thread pool and blocking synchronous clients that don't provide the full benefit of nonblocking I/O.

Synchronous methods block your thread's execution until the client receives a response from the service. Asynchronous methods return immediately, giving control back to the calling thread without waiting for a response.

Because an asynchronous method returns before a response is available, you need a way to get the response when it's ready. The methods for asynchronous client in 2.x of the AWS SDK for Java return *CompletableFuture objects* that allow you to access the response when it's ready.

Non-streaming operations

For non-streaming operations, asynchronous method calls are similar to synchronous methods. However, the asynchronous methods in the AWS SDK for Java return a CompletableFuture object that contains the results of the asynchronous operation *in the future*.

Call the `CompletableFuture` `whenComplete()` method with an action to complete when the result is available. `CompletableFuture` implements the `Future` interface, so you can also get the response object by calling the `get()` method.

The following is an example of an asynchronous operation that calls a Amazon DynamoDB function to get a list of tables, receiving a `CompletableFuture` that can hold a ListTablesResponse object. The action defined in the call to `whenComplete()` is done only when the asynchronous call is complete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;
```

```
import java.util.concurrent.CompletableFuture;
```

Code

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
                .region(region)
                .build();

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

        CompletableFuture<ListTablesResponse> response =
 client.listTables(ListTablesRequest.builder()
                .build());

        // Map the response to another CompletableFuture containing just the table names
        CompletableFuture<List<String>> tableNames =
 response.thenApply(ListTablesResponse::tableNames);

        // When future is complete (either successfully or in error) handle the response
        tableNames.whenComplete((tables, err) -> {
            try {
                if (tables != null) {
                    tables.forEach(System.out::println);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
            } finally {
                // Lets the application shut down. Only close the client when you are
 completely done with it.
                client.close();
            }
        });
        tableNames.join();
    }
}
```

The following code example shows you how to retrieve an Item from a table by using the Asynchronous client. Invoke the `getItem` method of the DynamoDbAsyncClient and pass it a GetItemRequest object with the table name and primary key value of the item you want. This is typically how you pass data that the operation requires. In this example, notice that a String value is passed.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
   public static void getItem(DynamoDbAsyncClient client, String tableName, String key,
 String keyVal) {

        HashMap<String, AttributeValue> keyToGet =
                new HashMap<String, AttributeValue>();

        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal).build());

        try {

            // Create a GetItemRequest instance
            GetItemRequest request = GetItemRequest.builder()
                    .key(keyToGet)
                    .tableName(tableName)
                    .build();

            // Invoke the DynamoDbAsyncClient object's getItem
            java.util.Collection<AttributeValue> returnedItem =
 client.getItem(request).join().item().values();

            // Convert Set to Map
            Map<String, AttributeValue> map =
 returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
            Set<String> keys = map.keySet();
            for (String sinKey : keys) {
                System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

See the complete example on GitHub.

Streaming operations

For streaming operations, you must provide an AsyncRequestBody to provide the content incrementally, or an AsyncResponseTransformer to receive and process the response.

The following example uploads a file to Amazon S3 asynchronously by using the `PutObject` operation.

Imports

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
 * To run this AWS code example, ensure that you have setup your development environment,
 including your AWS credentials.
 *
```

```
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncOps {

     public static void main(String[] args) {

         final String USAGE = "\n" +
                 "Usage:\n" +
                 "    S3AsyncOps <bucketName> <key> <path>\n\n" +
                 "Where:\n" +
                 "    bucketName - the name of the Amazon S3 bucket (for example, bucket1).
 \n\n" +
                 "    key - the name of the object (for example, book.pdf). \n" +
                 "    path - the local path to the file (for example, C:/AWS/book.pdf).
 \n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
             System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
                .region(region)
                .build();

        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        // Put the object into the bucket
        CompletableFuture<PutObjectResponse> future = client.putObject(objectRequest,
                AsyncRequestBody.fromFile(Paths.get(path))
        );
        future.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object uploaded. Details: " + resp);
                } else {
                    // Handle error
                    err.printStackTrace();
                }
            } finally {
                // Only close the client when you are completely done with it
                client.close();
            }
        });

        future.join();
    }
}
```

The following example gets a file from Amazon S3 asynchronously by using the `GetObject` operation.

Imports

```
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.nio.file.Paths;
import java.util.concurrent.CompletableFuture;
```

Code

```
/**
 * To run this AWS code example, ensure that you have setup your development environment,
 including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class S3AsyncStreamOps {

    public static void main(String[] args) {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    S3AsyncStreamOps <bucketName> <objectKey> <path>\n\n" +
                "Where:\n" +
                "    bucketName - the name of the Amazon S3 bucket (for example, bucket1).
 \n\n" +
                "    objectKey - the name of the object (for example, book.pdf). \n" +
                "    path - the local path to the file (for example, C:/AWS/book.pdf).
 \n" ;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
         }

        String bucketName = args[0];
        String objectKey = args[1];
        String path = args[2];

        Region region = Region.US_WEST_2;
        S3AsyncClient client = S3AsyncClient.builder()
                .region(region)
                .build();

        GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        CompletableFuture<GetObjectResponse> futureGet = client.getObject(objectRequest,
                AsyncResponseTransformer.toFile(Paths.get(path)));

        futureGet.whenComplete((resp, err) -> {
            try {
                if (resp != null) {
                    System.out.println("Object downloaded. Details: "+resp);
                } else {
                    err.printStackTrace();
                }
            } finally {
               // Only close the client when you are completely done with it
                client.close();
            }
```

```
        });
        futureGet.join();
    }
}
```

Advanced operations

The AWS SDK for Java 2.x uses Netty, an asynchronous event-driven network application framework, to handle I/O threads. The AWS SDK for Java 2.x creates an `ExecutorService` behind Netty, to complete the futures returned from the HTTP client request through to the Netty client. This abstraction reduces the risk of an application breaking the async process if developers choose to stop or sleep threads. By default, each asynchronous client creates a threadpool based on the number of processors and manages the tasks in a queue within the `ExecutorService`.

Advanced users can specify their thread pool size when creating an asynchronous client using the following option when building.

Code

```
S3AsyncClient clientThread = S3AsyncClient.builder()
  .asyncConfiguration(
    b -> b.advancedOption(SdkAdvancedAsyncClientOption
      .FUTURE_COMPLETION_EXECUTOR,
      Executors.newFixedThreadPool(10)
    )
  )
  .build();
```

To optimize performance, you can manage your own thread pool executor, and include it when configuring your client.

```
ThreadPoolExecutor executor = new ThreadPoolExecutor(50, 50,
    10, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(<custom_value>),
    new ThreadFactoryBuilder()
      .threadNamePrefix("sdk-async-response").build());

// Allow idle core threads to time out
executor.allowCoreThreadTimeOut(true);

S3AsyncClient clientThread = S3AsyncClient.builder()
  .asyncConfiguration(
    b -> b.advancedOption(SdkAdvancedAsyncClientOption
      .FUTURE_COMPLETION_EXECUTOR,
      executor
    )
  )
  .build();
```

If you prefer to not use a thread pool, at all, use *Runnable::run* instead of using a thread pool executor.

```
S3AsyncClient clientThread = S3AsyncClient.builder()
  .asyncConfiguration(
    b -> b.advancedOption(SdkAdvancedAsyncClientOption
        .FUTURE_COMPLETION_EXECUTOR,
        Runnable::run
      )
    )
```

```
    .build();
```

Using the DynamoDB Enhanced Client in the AWS SDK for Java 2.x

The DynamoDB enhanced client is a high-level library that is part of the AWS SDK for Java version 2 (v2). It offers a straightforward way to map client-side classes to DynamoDB tables. You define the relationships between tables and their corresponding model classes in your code. After you define those relationships, you can intuitively perform various create, read, update, or delete (CRUD) operations on tables or items in DynamoDB.

For information about how to use the DynamoDB Enhanced Client, refer to the section called "Mapping items in DynamoDB tables" (p. 131)

Working with HTTP/2 in the AWS SDK for Java

HTTP/2 is a major revision of the HTTP protocol. This new version has several enhancements to improve performance:

- Binary data encoding provides more efficient data transfer.
- Header compression reduces the overhead bytes downloaded by the client, helping get the content to the client sooner. This is especially useful for mobile clients that are already constrained on bandwidth.
- Bidirectional asynchronous communication (multiplexing) allows multiple requests and response messages between the client and AWS to be in flight at the same time over a single connection, instead of over multiple connections, which improves performance.

Developers upgrading to the latest SDKs will automatically use HTTP/2 when it's supported by the service they're working with. New programming interfaces seamlessly take advantage of HTTP/2 features and provide new ways to build applications.

The AWS SDK for Java 2.x features new APIs for event streaming that implement the HTTP/2 protocol. For examples of how to use these new APIs, see Working with Kinesis (p. 192).

Enabling SDK metrics for the AWS SDK for Java

With the AWS SDK for Java 2.x, you can collect metrics about the service clients in your application, analyze the output in Amazon CloudWatch, and then act on it.

By default, metrics collection is disabled in the SDK. This topic helps you to enable and configure it.

Prerequisites

Before you can enable and use metrics, you must complete the following steps:

- Complete the steps in *Setup* (p. 11).
- Configure your project dependencies (for example, in your `pom.xml` or `build.gradle` file) to use version `2.14.0` or later of the AWS SDK for Java.

To enabling publishing of metrics to CloudWatch, also include the artifactId `cloudwatch-metric-publisher` with the version number `2.14.0` or later in your project's dependencies.

For example:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
      <version>2.14.0</version>
   </dependency>
  </dependencies>
</project>
```

How to enable metrics collection

You can enable metrics in your application for a service client or on individual requests.

Enable metrics for a specific request

The following code snippet shows how to enable the CloudWatch metrics publisher for a request to Amazon DynamoDB. It uses the default metrics publisher configuration.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
DynamoDbClient ddb = DynamoDbClient.create();
ddb.listTables(ListTablesRequest.builder()
  .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
  .build());
```

Enable metrics for a specific service client

The following code snippet shows how to enable a CloudWatch metrics publisher with default settings for a service client.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
          .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
          .build();
```

The following snippet demonstrates how to use a custom configuration for the metrics publisher for a specific service client. The customizations include loading a specific credentials profile, specifying a different region than the service client, and customizing how often the publisher sends metrics to CloudWatch.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
    .cloudWatchClient(CloudWatchAsyncClient.builder()
                                            .region(Region.US_WEST_2)

 .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
                                            .build())
    .uploadFrequency(Duration.ofMinutes(5))
    .build();

DynamoDbClient ddb = DynamoDbClient.builder()
          .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
          .build();
```

What information is collected?

Metrics collection includes the following:

- Number of API requests, including whether they succeed or fail
- Information about the AWS services you call in your API requests, including exceptions returned
- The duration for various operations such as Marshalling, Signing, and HTTP requests
- HTTP client metrics, such as the number of open connections, the number of pending requests, and the name of the HTTP client used

> **Note**
> The metrics available vary by HTTP client.

For a complete list, see Service client metrics (p. 85).

How can I use this information?

You can use the metrics the SDK collects to monitor the service clients in your application. You can look at overall usage trends, identify anomalies, review service client exceptions returned, or to dig in to understand a particular issue. Using Amazon CloudWatch, you can also create alarms to notify you as soon as your application reaches a condition that you define.

For more information, see Using Amazon CloudWatch Metrics and Using Amazon CloudWatch Alarms in the Amazon CloudWatch User Guide.

Service client metrics

With the AWS SDK for Java 2.x, you can collect metrics from the service clients in your application and then publish (output) those metrics to Amazon CloudWatch.

These tables list the metrics that you can collect and any HTTP client usage requirement.

For more information about enabling and configuring metrics for the SDK, see Enabling SDK metrics (p. 83).

The terms used in the tables mean:

- Apache: the Apache-based HTTP client (`ApacheHttpClient`)
- Netty: the Netty-based HTTP client (`NettyNioAsyncHttpClient`)
- CRT: the AWS CRT-based HTTP client (`AwsCrtAsyncHttpClient`)

- Any: the collection of metric data does not depend on the HTTP client; this includes use of the URLConnection-based HTTP client (`UrlConnectionHttpClient`)

Metrics collected with each request

Metric name	Description	Type	HTTP client required
ApiCallDuration	The total time taken to finish a request (inclusive of all retries)	Duration	Any
ApiCallSuccessful	True if the API call was successful; false if not	Boolean	Any
CredentialsFetchDuration	The time taken to fetch AWS signing credentials for the request	Duration	Any
MarshallingDuration	The time taken to marshall the request	Duration	Any
OperationName	The name of the AWS API the request is made to	String	Any
RetryCount	Number of times the SDK retried the API call	Integer	Any
ServiceId	Service ID of the AWS service that the API request is made against	String	Any
TokenFetchDuration	The time taken to fetch token signing credentials for the request	Duration	Any

Metrics collected for each request attempt

Each API call that your application makes may take multiple attempts before responded with a success or failure. These metrics are collected for each attempt.

Metric name	Description	Type	HTTP client required
AvailableConcurrency	The number of remaining concurrent requests that can be supported by the HTTP client without needing to establish another connection	Integer	Apache, Netty, CRT
AwsExtendedRequestId	The extended request ID of the service request	String	Any

Metric name	Description	Type	HTTP client required
AwsRequestId	The request ID of the service request	String	Any
BackoffDelayDuration	The duration of time the SDK waited before this API call attempt	Duration	Any
ConcurrencyAcquireDuration	The time taken to acquire a channel from the connection pool	Duration	Apache, Netty, CRT
HttpClientName	The name of the HTTP being use for the request	String	Apache, Netty, CRT
HttpStatusCode	The status code returned with the HTTP response	Integer	Any
LeasedConcurrency	The number of request currently being executed by the HTTP client	Integer	Apache, Netty, CRT
LocalStreamWindowSize	The local HTTP/2 window size in bytes for the stream that this request was executed on	Integer	Netty
MarshallingDuration	The time it takes to marshall an SDK request to an HTTP request	Duration	Any
MaxConcurrency	The max number of concurrent requests supported by the HTTP client	Integer	Apache, Netty, CRT
PendingConcurrencyAcquires	The number of requests that are blocked, waiting for another TCP connection or a new stream to be available from the connection pool	Integer	Apache, Netty, CRT
RemoteStreamWindowSize	The remote HTTP/2 window size in bytes for the stream that this request was executed on	Integer	Netty

Metric name	Description	Type	HTTP client required
ServiceCallDuration	The time it takes to connect to the service, send the request, and receive the HTTP status code and header from the response	Duration	Any
SigningDuration	The time it takes to sign the HTTP request	Duration	Any
UnmarshallingDuration	The time it takes to unmarshall an HTTP response to an SDK response	Duration	Any

Retrieving paginated results using the AWS SDK for Java 2.x

Many AWS operations return paginated results when the response object is too large to return in a single response. In the AWS SDK for Java 1.0, the response contained a token you had to use to retrieve the next page of results. New in the AWS SDK for Java 2.x are autopagination methods that make multiple service calls to get the next page of results for you automatically. You only have to write code that processes the results. Additionally both types of methods have synchronous and asynchronous versions. See examples-asynchronous for more detail about asynchronous clients.

The following examples use Amazon S3 and Amazon DynamoDB operations to demonstrate the various methods of retrieving your data from paginated responses.

> **Note**
> These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in setup-credentials.

Synchronous pagination

These examples use the synchronous pagination methods for listing objects in an Amazon S3 bucket.

Iterate over pages

Build a ListObjectsV2Request and provide a bucket name. Optionally you can provide the maximum number of keys to retrieve at one time. Pass it to the S3Client's `listObjectsV2Paginator` method. This method returns a ListObjectsV2Iterable object, which is an `Iterable` of the ListObjectsV2Response class.

The first example demonstrates using the paginator object to iterate through all the response pages with the `stream` method. You can directly stream over the response pages, convert the response stream to a stream of S3Object content, and then process the content of the Amazon S3 object.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        // Process response pages
        listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() + " size =
 " + content.size()));
```

See the complete example on GitHub.

Iterate over objects

The following examples show ways to iterate over the objects returned in the response instead of the pages of the response.

Use a stream

Use the `stream` method on the response content to iterate over the paginated item collection.

Code

```
        // Helper method to work with paginated collection of items directly
        listRes.contents().stream()
                .forEach(content -> System.out.println(" Key: " + content.key() + " size =
 " + content.size()));
```

See the complete example on GitHub.

Use a for loop

Use a standard `for` loop to iterate through the contents of the response.

Code

```
        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " + content.size());
        }
```

See the complete example on GitHub.

Manual pagination

If your use case requires it, manual pagination is still available. Use the next token in the response object for the subsequent requests. Here's an example using a `while` loop.

Code

```
        ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse = s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }

            listObjectsReqManual = listObjectsReqManual.toBuilder()
                    .continuationToken(listObjResponse.nextContinuationToken())
                    .build();
        }
```

See the complete example on GitHub.

Asynchronous pagination

These examples use the asynchronous pagination methods for listing tables in DynamoDB. A manual pagination example is available in the basics-async topic.

Iterate over pages of table names

First, create an asynchronous DynamoDB client. Then, call the `listTablesPaginator` method to get a ListTablesPublisher. This is an implementation of the reactive streams Publisher interface. To learn more about the reactive streams model, see the Reactive Streams Github repo.

Call the `subscribe` method on the ListTablesPublisher and pass a subscriber implementation. In this example, the subscriber has an `onNext` method that requests one item at a time from the publisher. This is the method that is called repeatedly until all pages are retrieved. The `onSubscribe` method calls the `Subscription.request` method to initiate requests for data from the publisher. This method must be called to start getting data from the publisher. The `onError` method is triggered if an error occurs while retrieving data. Finally, the `onComplete` method is called when all pages have been requested.

Use a subscriber

Imports

```
import java.util.List;
```

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;

import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;
import io.reactivex.Flowable;
import reactor.core.publisher.Flux;
```

Code

First create an async client

```
       // Creates a default client with credentials and regions loaded from the
environment
       final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

       ListTablesRequest listTablesRequest = ListTablesRequest.builder().limit(3).build();
```

Then use Subscriber to get results.

```
       // Or subscribe method should be called to create a new Subscription.
       // A Subscription represents a one-to-one life-cycle of a Subscriber subscribing to
a Publisher.
       publisher.subscribe(new Subscriber<ListTablesResponse>() {
           // Maintain a reference to the subscription object, which is required to
request data from the publisher
           private Subscription subscription;

           @Override
           public void onSubscribe(Subscription s) {
               subscription = s;
               // Request method should be called to demand data. Here we request a single
page
               subscription.request(1);
           }

           @Override
           public void onNext(ListTablesResponse response) {
               response.tableNames().forEach(System.out::println);
               // Once you process the current page, call the request method to signal
that you are ready for next page
               subscription.request(1);
           }

           @Override
           public void onError(Throwable t) {
               // Called when an error has occurred while processing the requests
           }

           @Override
           public void onComplete() {
               // This indicates all the results are delivered and there are no more pages
left
           }
```

See the complete example on GitHub.

Use a for loop

Use a `for` loop to iterate through the pages for simple use cases when creating a new subscriber might be too much overhead. The response publisher object has a `forEach` helper method for this purpose.

Code

```
        ListTablesPublisher publisher = asyncClient.listTablesPaginator(listTablesRequest);

        // Use a for-loop for simple use cases
        CompletableFuture<Void> future = publisher.subscribe(response ->
response.tableNames()

.forEach(System.out::println));
```

See the complete example on GitHub.

Iterate over table names

The following examples show ways to iterate over the objects returned in the response instead of the pages of the response. Similar to the synchronous result, the asynchronous result class has a method to interact with the underlying item collection. The return type of the convenience method is a publisher that can be used to request items across all pages.

Use a subscriber

Code

First create an async client

```
        System.out.println("running AutoPagination - iterating on item collection...\n");

        // Creates a default client with credentials and regions loaded from the
environment
        final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

        ListTablesRequest listTablesRequest = ListTablesRequest.builder().limit(3).build();
```

Then use Subscriber to get results.

```
        // Use subscriber
        publisher.subscribe(new Subscriber<String>() {
            private Subscription subscription;

            @Override
            public void onSubscribe(Subscription s) {
                subscription = s;
                subscription.request(1);
            }

            @Override
            public void onNext(String tableName) {
                System.out.println(tableName);
                subscription.request(1);
            }

            @Override
            public void onError(Throwable t) { }

            @Override
```

```
            public void onComplete() { }
```

See the complete example on GitHub.

Use a for loop

Use the `forEach` convenience method to iterate through the results.

Code

```
        // Use forEach
        CompletableFuture<Void> future = publisher.subscribe(System.out::println);
        future.get();
```

See the complete example on GitHub.

Use third-party library

You can use other third party libraries instead of implementing a custom subscriber. This example demonstrates using the RxJava implementation but any library that implements the reactive stream interfaces can be used. See the RxJava wiki page on Github for more information on that library.

To use the library, add it as a dependency. If using Maven, the example shows the POM snippet to use.

POM Entry

```
<dependency>
      <groupId>io.reactivex.rxjava2</groupId>
      <artifactId>rxjava</artifactId>
      <version>2.2.21</version>
</dependency>
```

Imports

```
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;

import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.paginators.ListTablesPublisher;
import io.reactivex.Flowable;
import reactor.core.publisher.Flux;
```

Code

```
        System.out.println("running AutoPagination - using third party subscriber...\n");

        DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
        ListTablesPublisher publisher =
 asyncClient.listTablesPaginator(ListTablesRequest.builder()

 .build());
```

```
        // The Flowable class has many helper methods that work with any reactive streams
 compatible publisher implementation
        List<String> tables = Flowable.fromPublisher(publisher)
                                      .flatMapIterable(ListTablesResponse::tableNames)
                                      .toList()
                                      .blockingGet();
        System.out.println(tables);
```

AWS CRT-based S3 client

The AWS CRT-based S3 client—built on top of the AWS Common Runtime (CRT)—is an alternative S3 asynchronous client. It transfers objects to and from Amazon Simple Storage Service (Amazon S3) with enhanced performance and reliability by automatically using Amazon S3's multipart upload API and byte-range fetches.

The AWS CRT-based S3 client improves transfer reliability in case there is a network failure. Reliability is improved by retrying individual failed parts of a file transfer without restarting the transfer from the beginning.

In addition, the AWS CRT-based S3 client offers enhanced connection pooling and Domain Name System (DNS) load balancing, which also improves throughput.

You can use the AWS CRT-based S3 client in place of the SDK's standard S3 asynchronous client and take advantage of its improved throughput right away.

AWS CRT-based components in the SDK

The AWS CRT-based *S3* client, described in this topic, and the AWS CRT-based *HTTP* client are different components in the SDK.

The **AWS CRT-based S3 client** is an implementation of the S3AsyncClient interface and is used for working with the Amazon S3 service. It is an alternative to the Java-based implementation of the `S3AsyncClient` interface and offers several benefits.

The AWS CRT-based HTTP client (p. 62) is an implementation of the SdkAsyncHttpClient interface and is used for general HTTP communication. It is an alternative to the Netty implementation of the `SdkAsyncHttpClient` interface and offers several advantages.

Although both components use libraries from the AWS Common Runtime, the AWS CRT-based S3 client uses the aws-c-s3 library and supports the S3 multipart upload API features. Since the AWS CRT-based HTTP client is meant for general purpose use, it does not support the S3 multipart upload API features.

Add dependencies to use the AWS CRT-based S3 client

To use the AWS CRT-based S3 client, add the following two dependencies to your Maven project file. The example shows the minimum versions to use. Search the Maven central repository for the most recent versions of the s3 and aws-crt artifacts.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.19.0</version>
</dependency>
```

```
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.20.3</version>
</dependency>
```

Create an instance of the AWS CRT-based S3 client

Create an instance of the AWS CRT-based S3 client with default settings as shown in the following code snippet.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

To configure the client, use the AWS CRT client builder. You can switch from the standard S3 asynchronous client to AWS CRT-based client by changing the builder method.

```
S3AsyncClient s3AsyncClient =
        S3AsyncClient.crtBuilder()
                      .credentialsProvider(DefaultCredentialsProvider.create())
                      .region(Region.US_WEST_2)
                      .targetThroughputInGbps(20.0)
                      .minimumPartSizeInBytes(8 * MB)
                      .build();
```

Note
Some of the settings in the standard builder might not be currently supported in the AWS CRT client builder. Get the standard builder by calling `S3AsyncClient#builder()`.

Use the AWS CRT-based S3 client

Use the AWS CRT-based S3 client to call Amazon S3 API operations. The following example demonstrates the PutObject and GetObject operations available through the AWS SDK for Java.

```
S3AsyncClient s3Client = S3AsyncClient.crtCreate();

// Upload a local file to Amazon S3.
PutObjectResponse response =
      s3Client.putObject(req -> req.bucket(<BUCKET_NAME>)
                                   .key(<KEY_NAME>),
                         AsyncRequestBody.fromFile(Paths.get(<FILE_NAME>)))
              .join();

// Download an object from Amazon S3 to a local file.
GetObjectResponse response =
      s3Client.getObject(req -> req.bucket(<BUCKET_NAME>)
                                   .key(<KEY_NAME>),
                         AsyncResponseTransformer.toFile(Paths.get(<FILE_NAME>)))
              .join();
```

Amazon S3 Transfer Manager

The Amazon S3 Transfer Manager is an open source, high level file transfer utility for the AWS SDK for Java 2.x. Use it to transfer files and directories to and from Amazon Simple Storage Service (Amazon S3).

When built on top of the the section called "AWS CRT-based S3 client" (p. 94), the S3 Transfer Manager can take advantage of performance improvements such as multipart upload API and byte-range fetches.

With the S3 Transfer Manager, you can also monitor a transfer's progress in real time and pause the transfer for later execution.

Get started

Add dependencies to your build file

To use the S3 Transfer Manager with enhanced performance based on the AWS CRT-based S3 client, configure your build file with the following dependencies.

- Use version *2.19.1* or higher of the SDK for Java 2.x.
- Add the `s3-transfer-manager` artifact as a dependency.
- Add the `aws-crt` artifact as a dependency at version *0.20.3* or higher.

The following code example shows how to configure your project dependencies for Maven.

```
<project>
   <properties>
     <aws.sdk.version>2.19.1</aws.sdk.version>
  </properties>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3-transfer-manager</artifactId>
   </dependency>
   <dependency>
      <groupId>software.amazon.awssdk.crt</groupId>
      <artifactId>aws-crt</artifactId>
      <version>0.20.3</version>
   </dependency>
  </dependencies>
</project>
```

Search the Maven central repository for the most recent versions of the s3-transfer-manager and aws-crt artifacts.

Create an instance of the S3 Transfer Manager

The following snippet shows how to create a S3TransferManager instance with default settings.

```
        S3TransferManager transferManager = S3TransferManager.create();
```

The following example shows how to configure a S3 Transfer Manager with custom settings. In this example, a AWS CRT-based S3AsyncClient (p. 94) instance is used as the underlying client for the S3 Transfer Manager.

```
        S3AsyncClient s3AsyncClient =
            S3AsyncClient.crtBuilder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .targetThroughputInGbps(20.0)
                .minimumPartSizeInBytes(8 * MB)
                .build();

        S3TransferManager transferManager =
            S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();
```

Note
If the `aws-crt` dependency is not included in the build file, the S3 Transfer Manager is built on top of the standard S3 asynchronous client used in the SDK for Java 2.x.

Upload a file to an S3 bucket

To upload a file to Amazon S3 using the S3 Transfer Manager, pass an `UploadFileRequest` object to the `S3TransferManager`'s uploadFile method.

The FileUpload object returned from the `uploadFile` method represents the upload process. After the request finishes, the CompletedFileUpload object contains information about the upload.

The following example shows a file upload example along with the optional use of a LoggingTransferListener, which logs the progress of the upload.

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

Code

```
    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, String filePath) {
        UploadFileRequest uploadFileRequest =
            UploadFileRequest.builder()
                .putObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())
                .source(Paths.get(filePath))
                .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
```

```
        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Download a file from an S3 bucket

To download an object from an S3 bucket using the S3 Transfer Manager, build a DownloadFileRequest object and pass it to the downloadFile method.

The FileDownload object returned by the `S3TransferManager`'s `downloadFile` method represents the file transfer. After the download completes, the CompletedFileDownload contains access to information about the download.

The following example also shows a download example plus the optional use of a LoggingTransferListener, which logs the progress of the download.

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.UUID;
```

Code

```
    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest =
            DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
        logger.info("Content length [{}]", downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Copy an Amazon S3 object to another bucket

To begin the copy of an object from an S3 bucket to another bucket, create a basic CopyObjectRequest instance.

Next, wrap the basic `CopyObjectRequest` in a CopyRequest that can be used by the S3 Transfer Manager.

The `Copy` object returned by the `S3TransferManager`'s `copy` method represents the copy process. After the copy process completes, the CompletedCopy object contains details about the response.

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;
```

Code

```
    public String copyObject(S3TransferManager transferManager, String bucketName,
                             String key, String destinationBucket, String destinationKey){
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
            .sourceBucket(bucketName)
            .sourceKey(key)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        CopyRequest copyRequest = CopyRequest.builder()
            .copyObjectRequest(copyObjectRequest)
            .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

Note
Cross-Region copies are not currently supported.

Upload a local directory to an S3 bucket

To upload a local directory to an S3 bucket, start by calling the uploadDirectory method of the `S3TransferManager` instance, passing in an UploadDirectoryRequest.

The DirectoryUpload object represents the upload process, which generates a CompletedDirectoryUpload when the request completes. The `CompleteDirectoryUpload` object contains information about the results of the transfer, including which files failed to transfer.

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URL;
```

```
import java.nio.file.Paths;
import java.util.UUID;
```

Code

```
    public Integer uploadDirectory(S3TransferManager transferManager,
                                   String sourceDirectory, String bucketName){
        DirectoryUpload directoryUpload =
            transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
 directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers().forEach(fail ->
            logger.warn("Object [{}] failed to transfer", fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

Download S3 bucket objects to a local directory

To download the objects in an S3 bucket to a local directory, begin by calling the downloadDirectory method of the Transfer Manager, passing in a DownloadDirectoryRequest.

The DirectoryDownload object represents the download process, which generates a CompletedDirectoryDownload when the request completes. The `CompleteDirectoryDownload` object contains information about the results of the transfer, including which files failed to transfer.

Imports

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

Code

```
    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
                                              String destinationPath, String bucketName) {
        DirectoryDownload directoryDownload =
            transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPath))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();
```

```
        completedDirectoryDownload.failedTransfers().forEach(fail ->
            logger.warn("Object [{}] failed to transfer", fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

Using waiters in the AWS SDK for Java 2.x

The waiters utility of the AWS SDK for Java 2.x enables you to validate that AWS resources are in a specified state before performing operations on those resources.

A *waiter* is an abstraction used to poll AWS resources, such as DynamoDB tables or Amazon S3 buckets, until a desired state is reached (or until a determination is made that the resource won't ever reach the desired state). Instead of writing logic to continuously poll your AWS resources, which can be cumbersome and error-prone, you can use waiters to poll a resource and have your code continue to run after the resource is ready.

Prerequisites

Before you can use waiters in a project with the AWS SDK for Java, you must complete the steps in Setting up the AWS SDK for Java 2.x (p. 11).

You must also configure your project dependencies (for example, in your `pom.xml` or `build.gradle` file) to use version `2.15.0` or later of the AWS SDK for Java.

For example:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
</project>
```

Using waiters

To instantiate a waiters object, first create a service client. Set the service client's `waiter()` method as the value of the waiter object. Once the waiter instance exists, set its response options to execute the appropriate code.

Synchronous programming

The following code snippet shows how to wait for a DynamoDB table to exist and be in an **ACTIVE** state.

```
DynamoDbClient dynamo = DynamoDbClient.create();
DynamoDbWaiter waiter = dynamo.waiter();
```

```
WaiterResponse<DescribeTableResponse> waiterResponse =
  waiter.waitUntilTableExists(r -> r.tableName("myTable"));

// print out the matched response with a tableStatus of ACTIVE
waiterResponse.matched().response().ifPresent(System.out::println);
```

Asynchronous programming

The following code snippet shows how to wait for a DynamoDB table to no longer exist.

```
DynamoDbAsyncClient asyncDynamo = DynamoDbAsyncClient.create();
DynamoDbAsyncWaiter asyncWaiter = asyncDynamo.waiter();

CompletableFuture<WaiterResponse<DescribeTableResponse>> waiterResponse =
          asyncWaiter.waitUntilTableNotExists(r -> r.tableName("myTable"));

waiterResponse.whenComplete((r, t) -> {
  if (t == null) {
   // print out the matched ResourceNotFoundException
   r.matched().exception().ifPresent(System.out::println);
  }
}).join();
```

Configuring waiters

You can customize the configuration for a waiter by using the `overrideConfiguration()` on its builder. For some operations, you can apply a custom configuration when you make the request.

Configure a waiter

The following code snippet shows how to override the configuration on a waiter.

```
// sync
DynamoDbWaiter waiter =
   DynamoDbWaiter.builder()
          .overrideConfiguration(b -> b.maxAttempts(10))
          .client(dynamoDbClient)
          .build();
// async
DynamoDbAsyncWaiter asyncWaiter =
   DynamoDbAsyncWaiter.builder()
          .client(dynamoDbAsyncClient)
          .overrideConfiguration(o -> o.backoffStrategy(
               FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
          .scheduledExecutorService(Executors.newScheduledThreadPool(3))
          .build();
```

Override configuration for a specific request

The following code snippet shows how to override the configuration for a waiter on a per-request basis. Note that only some operations have customizable configurations.

```
waiter.waitUntilTableNotExists(b -> b.tableName("myTable"),
               o -> o.maxAttempts(10));

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
                 o -> o.waitTimeout(Duration.ofMinutes(1)));
```

Code examples

For a complete example using waiters with DynamoDB, see CreateTable.java in the AWS Code Examples Repository.

For a complete example using waiters with Amazon S3, see S3BucketOps.java in the AWS Code Examples Repository.

Code examples for the AWS SDK for Java 2.x

This section provides programming examples you can use with the AWS SDK for Java 2.x for specific features, use cases, and AWS services.

Find the source code for these examples and others in the AWS documentation code examples repository on GitHub.

To propose a new code example for the AWS documentation team to consider producing, create a new request. The team is looking to produce code examples that cover broader scenarios and use cases, versus simple code snippets that cover only individual API calls. For instructions, see the "Proposing new code examples" section in the Readme on GitHub.

Topics

- Guided code examples for the AWS SDK for Java 2.x (p. 104)
- SDK for Java 2.x code examples (p. 222)

Guided code examples for the AWS SDK for Java 2.x

This section provides guided code examples you can use to build applications that use AWS services.

Find the source code for these examples and others in the AWS documentation code examples repository on GitHub.

To propose a new code example for the AWS documentation team to consider producing, create a new request. The team is looking to produce code examples that cover broader scenarios and use cases, versus simple code snippets that cover only individual API calls. For instructions, see the "Proposing new code examples" section in the Readme on GitHub.

Topics

- AWS database services and AWS SDK for Java 2.x (p. 105)
- Working with Amazon S3 (p. 106)
- Working with DynamoDB (p. 120)
- Working with Amazon EC2 (p. 141)
- Working with IAM (p. 160)
- Working with Amazon Athena (p. 177)
- Working with CloudWatch (p. 177)
- Working with AWS CloudTrail (p. 187)
- Working with Amazon Cognito (p. 187)
- Working with Amazon Kinesis Data Firehose (p. 192)
- Working with Amazon Forecast (p. 192)

- Working with Kinesis (p. 192)
- Invoke, list, and delete AWS Lambda functions (p. 198)
- Working with AWS Elemental MediaConvert (p. 201)
- Working with AWS Elemental MediaStore (p. 201)
- Working with AWS Migration Hub (p. 201)
- Working with Amazon Pinpoint (p. 201)
- Working with Amazon Polly (p. 208)
- Working with Amazon SageMaker (p. 209)
- Working with Amazon Simple Notification Service (p. 209)
- Working with Amazon Simple Queue Service (p. 213)
- Working with AWS Systems Manager (p. 217)
- Working with Amazon Simple Workflow Service (p. 218)
- Working with Amazon Transcribe (p. 218)
- Working with Amazon Translate (p. 222)
- Working with Amazon WorkDocs (p. 222)

AWS database services and AWS SDK for Java 2.x

AWS offers several database types: relational, key-value, in-memory, document, and several others. The SDK for Java 2.x support varies depending the nature of the database service in AWS.

Some database services, for example Amazon DynamoDB service, have web service APIs to manage the AWS resource (database) as well as web service APIs to interact with the data. In the SDK for Java 2.x these types of services have dedicated service clients, for example DynamoDBClient.

Other database services have web service APIs that interact with the resource, such the Amazon DocumentDB API (for cluster, instance and resource management), but do not have a web service API for working with the data. The SDK for Java 2.x has a corresponding DocDbClient interface for working with the resource. However, you need another Java API, such as MongoDB for Java to work with the data.

Use the examples below to learn how you use the SDK for Java 2.x service clients with the different types of databases.

Amazon DynamoDB examples

Working with the data	Working with the database
SDK service client: DynamoDBClient Example: React/Spring REST application using DynamoDB Examples: Several DynamoDB examples SDK service client: DynamoDBEnhancedClient Example: React/Spring REST application using DynamoDB Examples: Several DynamoDB examples (names starting with 'Enhanced")	SDK service client: DynamoDBClient Examples: CreateTable, ListTables, DeleteTable

See additional DynamoDB examples (p. 120) in the guided code examples section of this guide.

Amazon RDS examples

Working with the data	Working with the database
Non-SDK API: JDBC, database-specific SQL flavor	SDK service client: RdsClient
Example: React/Spring REST application using MySQL	Examples: Several RdsClient examples

Amazon Redshift examples

Working with the data	Working with the database
SDK service client: RedshiftDataClient	SDK service client: RedshiftClient
Examples: Several RedshiftDataClient examples	Examples: Several RedshiftClient examples
Example: React/Spring REST application using RedshiftDataClient	

Amazon Aurora Serverless v1 examples

Working with the data	Working with the database
SDK service client: RdsDataClient	SDK service client: RdsClient
Example: React/Spring REST application using RdsDataClient	Examples: Several RdsClient examples

Amazon DocumentDB examples

Working with the data	Working with the database
Non-SDK API: MongoDB-specific Java library (for example MongoDB for Java)	SDK service client: DocDbClient
Examples: DocumentDB (Mongo) Developer Guide (select 'Java' tab)	

Working with Amazon S3

This section provides examples of programming with Amazon Simple Storage Service (S3) using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Note

From version 2.18.x and onward, the AWS SDK for Java 2.x uses virtual host-style addressing when including an endpoint override, as long as the bucket name is a valid DNS label.
Call the `forcePathStyle` method with `true` in your client builder to force the client to use path-style addressing for buckets.
The following example shows a service client configured with an endpoint override and using path-style addressing.

```
S3Client client = S3Client.builder()
                          .region(Region.US_WEST_2)
                          .endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
                          .forcePathStyle(true)
                          .build();
```

Topics

- Creating, listing, and deleting Amazon S3 buckets (p. 107)
- Working with Amazon S3 objects (p. 111)
- Working with Amazon S3 presigned URLs (p. 117)
- Working with S3 Glacier (p. 120)

Creating, listing, and deleting Amazon S3 buckets

Every object (file) in Amazon S3 must reside within a *bucket*. A bucket represents a collection (container) of objects. Each bucket must have a unique *key* (name). For detailed information about buckets and their configuration, see Working with Amazon S3 Buckets in the Amazon Simple Storage Service User Guide.

Note

Best Practice
We recommend that you enable the AbortIncompleteMultipartUpload lifecycle rule on your Amazon S3 buckets.
This rule directs Amazon S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When the set time limit is exceeded, Amazon S3 aborts the upload and then deletes the incomplete upload data.
For more information, see Lifecycle Configuration for a Bucket with Versioning in the Amazon Simple Storage Service User Guide.

Note

These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in the section called "Set up single sign-on access for the SDK" (p. 13).

Topics

- Create a bucket (p. 107)
- List the buckets (p. 108)
- Delete a bucket (p. 109)

Create a bucket

Build a CreateBucketRequest and provide a bucket name. Pass it to the S3Client's `createBucket` method. Use the `S3Client` to do additional operations such as listing or deleting buckets as shown in later examples.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
        ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();
```

Make a Create Bucket Request.

```
     // Create a bucket by using a S3Waiter object
    public static void createBucket( S3Client s3Client, String bucketName) {

        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName +" is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

List the buckets

Build a ListBucketsRequest. Use the S3Client's `listBuckets` method to retrieve the list of buckets. If the request succeeds a ListBucketsResponse is returned. Use this response object to retrieve the list of buckets.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
        ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();
```

Make a List Buckets Request.

```
        // List buckets
        ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
        ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
        listBucketsResponse.buckets().stream().forEach(x -> System.out.println(x.name()));
```

See the complete example on GitHub.

Delete a bucket

Before you can delete an Amazon S3 bucket, you must ensure that the bucket is empty or the service will return an error. If you have a versioned bucket, you must also delete any versioned objects that are in the bucket.

Topics

- Delete objects in a bucket (p. 109)
- Delete an empty bucket (p. 110)

Delete objects in a bucket

Build a ListObjectsV2Request and use the S3Client's `listObjects` method to retrieve the list of objects in the bucket. Then use the `deleteObject` method on each object to delete it.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

Code

First create an S3Client.

```
        ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .credentialsProvider(credentialsProvider)
                .build();
```

Delete all objects in the bucket.

```
    public static void deleteObjectsInBucket (S3Client s3, String bucket) {

        try {
            // To delete a bucket, all the objects in the bucket must be deleted first.
            ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
                    .bucket(bucket)
                    .build();
            ListObjectsV2Response listObjectsV2Response;

            do {
                listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
                for (S3Object s3Object : listObjectsV2Response.contents()) {
                    DeleteObjectRequest request = DeleteObjectRequest.builder()
                            .bucket(bucket)
                            .key(s3Object.key())
                            .build();
                    s3.deleteObject(request);
                }
            } while (listObjectsV2Response.isTruncated());
```

See the complete example on GitHub.

Delete an empty bucket

Build a DeleteBucketRequest with a bucket name and pass it to the S3Client's `deleteBucket` method.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

Code

First create an S3Client.

```
        ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();
```

Delete the bucket.

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        s3.deleteBucket(deleteBucketRequest);
        s3.close();
```

See the complete example on GitHub.

Working with Amazon S3 objects

An Amazon S3 object represents a file or collection of data. Every object must be contained in a bucket (p. 107).

> **Note**
> Best Practice
> We recommend that you enable the AbortIncompleteMultipartUpload lifecycle rule on your Amazon S3 buckets.
> This rule directs Amazon S3 to abort multipart uploads that don't complete within a specified number of days after being initiated. When the set time limit is exceeded, Amazon S3 aborts the upload and then deletes the incomplete upload data.
> For more information, see Lifecycle Configuration for a Bucket with Versioning in the Amazon Simple Storage Service User Guide.

> **Note**
> These code snippets assume that you understand the material in basics, and have configured default AWS credentials using the information in the section called "Set up single sign-on access for the SDK" (p. 13).

Topics

- Upload an object (p. 111)
- Upload objects in multiple parts (p. 112)
- Download an object (p. 114)
- Delete an object (p. 115)
- Copy an object (p. 115)
- List objects (p. 116)

Upload an object

Build a PutObjectRequest and supply a bucket name and key name. Then use the S3Client's `putObject` method with a RequestBody that contains the object content and the `PutObjectRequest` object. *The bucket must exist, or the service will return an error.*

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
        Region region = Region.US_WEST_2;
        s3 = S3Client.builder()
                .region(region)
                .build();

        createBucket(s3, bucketName, region);

        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.putObject(objectRequest,
 RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

See the complete example on GitHub.

Upload objects in multiple parts

Use the S3Client's `createMultipartUpload` method to get an upload ID. Then use the `uploadPart` method to upload each part. Finally, use the S3Client's `completeMultipartUpload` method to tell Amazon S3 to merge all the uploaded parts and finish the upload operation.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
       // First create a multipart upload and get the upload id
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();

       CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
       String uploadId = response.uploadId();
       System.out.println(uploadId);

       // Upload all the different parts of the object
       UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
               .bucket(bucketName)
                .key(key)
               .uploadId(uploadId)
               .partNumber(1).build();

       String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB))).eTag();

       CompletedPart part1 = CompletedPart.builder().partNumber(1).eTag(etag1).build();

       UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
               .uploadId(uploadId)
               .partNumber(2).build();
       String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB))).eTag();
       CompletedPart part2 = CompletedPart.builder().partNumber(2).eTag(etag2).build();


       // Finally call completeMultipartUpload operation to tell S3 to merge all uploaded
       // parts and finish the multipart operation.
       CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
               .parts(part1, part2)
               .build();
```

```
        CompleteMultipartUploadRequest completeMultipartUploadRequest =
                CompleteMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .multipartUpload(completedMultipartUpload)
                        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
```

See the complete example on GitHub.

Download an object

Build a GetObjectRequest and supply a bucket name and key name. Use the S3Client's `getObject` method, passing it the `GetObjectRequest` object and a `ResponseTransformer` object. The `ResponseTransformer` creates a response handler that writes the response content to the specified file or stream.

The following example specifies a file name to write the object content to.

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.getObject(getObjectRequest);
```

See the complete example on GitHub.

Delete an object

Build a DeleteObjectRequest and supply a bucket name and key name. Use the S3Client's `deleteObject` method, and pass it the name of a bucket and object to delete. *The specified bucket and object key must exist, or the service will return an error.*

Imports

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Random;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
```

Code

```
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.deleteObject(deleteObjectRequest);
```

See the complete example on GitHub.

Copy an object

Build a CopyObjectRequest and supply a bucket name that the object is coped into, a URL encoded string value (see the URLEncoder.encode method), and the key name of the object. Use the S3Client's `copyObject` method, and pass the CopyObjectRequest object. *The specified bucket and object key must exist, or the service will return an error.*

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
```

Code

```
    public static String copyBucketObject (S3Client s3, String fromBucket, String
 objectKey, String toBucket) {

        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

See the complete example on GitHub.

List objects

Build a ListObjectsRequest and supply the bucket name. Then invoke the S3Client's `listObjects` method and pass the ListObjectsRequest object. This method returns a ListObjectsResponse that contains all of the objects in the bucket. You can invoke this object's *contents* method to get a list of objects. You can iterate through this list to display the objects, as shown in the following code example.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

Code

```
    public static void listBucketObjects(S3Client s3, String bucketName ) {

        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                .builder()
                .bucket(bucketName)
                .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    //convert bytes to kbs.
    private static long calKb(Long val) {
        return val/1024;
    }
```

See the complete example on GitHub.

Working with Amazon S3 presigned URLs

You can use a S3Presigner object to sign an Amazon S3 `SdkRequest` so that it's executed without requiring authentication on the part of the caller. For example, assume Alice has access to an S3 object, and she wants to temporarily share access to that object with Bob. Alice can generate a pre-signed GetObjectRequest object to secure share with Bob so that he can download the object without requiring access to Alice's credentials.

Topics

- Generate a Presigned URL and Upload an Object (p. 117)
- Get a Presigned Object (p. 118)

Generate a Presigned URL and Upload an Object

Build a S3Presigner object that represents the client object. Next create a PresignedPutObjectRequest object that can be executed at a later time without requiring additional signing or authentication. When you create this object, you can specify the bucket name and the key name. In addition, you can also specify the time in minutes that the bucket can be accessed without using credentials by invoking the `signatureDuration` method (as shown in the following code example).

You can use the PresignedPutObjectRequest object to obtain the URL by invoking its `url` method.

Imports

```
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.time.Duration;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;
```

Code

The following Java code example uploads content to a presigned S3 bucket.

```
    public static void signBucket(S3Presigner presigner, String bucketName, String keyName)
 {
```

```
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author","Mary Doe");
            metadata.put("version","1.0.0.0");

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .contentType("text/plain")
                .metadata(metadata)
                .build();

            PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10))
                .putObjectRequest(objectRequest)
                .build();

            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
            System.out.println("Presigned URL to upload a file to: " +
presignedRequest.url());
            System.out.println("Which HTTP method needs to be used when uploading a file: "
+ presignedRequest.httpRequest().method());

            // Upload content to the Amazon S3 bucket by using this URL.
            URL url = presignedRequest.url();

            // Create the connection and use it to upload the new object.
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type","text/plain");
            connection.setRequestProperty("x-amz-meta-author","Mary Doe");
            connection.setRequestProperty("x-amz-meta-version","1.0.0.0");
            connection.setRequestMethod("PUT");
            OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
            out.write("This text was uploaded as an object by using a presigned URL.");
            out.close();

            connection.getResponseCode();
            System.out.println("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
```

See the complete example on GitHub.

Get a Presigned Object

Build a S3Presigner object that represents the client object. Next, create a GetObjectRequest object and specify the bucket name and key name. In addition, create a GetObjectPresignRequest object that can be executed at a later time without requiring additional signing or authentication. When you create this object, you can specify the time in minutes that the bucket can be accessed without using credentials by invoking the `signatureDuration` method (as shown in the following code example).

Invoke the `presignGetObject` method that belongs to the S3Presigner object to create a PresignedGetObjectRequest object. You can invoke this object's `url` method to obtain the URL to use. Once you have the URL, you can use standard HTTP Java logic to read the contents of the bucket, as shown in the following Java code example.

Imports

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;
```

Code

The following Java code example reads content from a presigned S3 bucket.

```
      public static void getPresignedUrl(S3Presigner presigner, String bucketName, String
 keyName ) {

          try {
              GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                  .bucket(bucketName)
                  .key(keyName)
                  .build();

              GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                  .signatureDuration(Duration.ofMinutes(60))
                  .getObjectRequest(getObjectRequest)
                  .build();

              PresignedGetObjectRequest presignedGetObjectRequest =
 presigner.presignGetObject(getObjectPresignRequest);
              String theUrl = presignedGetObjectRequest.url().toString();
              System.out.println("Presigned URL: " + theUrl);
              HttpURLConnection connection = (HttpURLConnection)
 presignedGetObjectRequest.url().openConnection();
              presignedGetObjectRequest.httpRequest().headers().forEach((header, values) -
> {
               values.forEach(value -> {
                   connection.addRequestProperty(header, value);
                 });
              });

              // Send any request payload that the service needs (not needed when
 isBrowserExecutable is true).
              if (presignedGetObjectRequest.signedPayload().isPresent()) {
                  connection.setDoOutput(true);

              try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                   OutputStream httpOutputStream = connection.getOutputStream()) {
                   IoUtils.copy(signedPayload, httpOutputStream);
              }
          }

          // Download the result of executing the request.
          try (InputStream content = connection.getInputStream()) {
              System.out.println("Service returned response: ");
              IoUtils.copy(content, System.out);
          }
```

```
        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
```

See the complete example on GitHub.

Working with S3 Glacier

S3 Glacier is an extremely low-cost storage service that provides secure, durable, and flexible storage for data backup and archival. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with DynamoDB

This section provides examples that show you how to program DynamoDB by using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Working with tables in DynamoDB (p. 120)
- Working with items in DynamoDB (p. 126)
- Mapping items in DynamoDB tables (p. 131)

Working with tables in DynamoDB

Tables are the containers for all items in a DynamoDB database. Before you can add or remove data from DynamoDB, you must create a table.

For each table, you must define:

- A table *name* that is unique for your account and region.
- A *primary key* for which every value must be unique; no two items in your table can have the same primary key value.

  A primary key can be *simple*, consisting of a single partition (HASH) key, or *composite*, consisting of a partition and a sort (RANGE) key.

  Each key value has an associated *data type*, enumerated by the ScalarAttributeType class. The key value can be binary (B), numeric (N), or a string (S). For more information, see Naming Rules and Data Types in the Amazon DynamoDB Developer Guide.
- *Provisioned throughput* are values that define the number of reserved read/write capacity units for the table.

  > **Note**
  > Amazon DynamoDB pricing is based on the provisioned throughput values that you set on your tables, so reserve only as much capacity as you think you'll need for your table.

  Provisioned throughput for a table can be modified at any time, so you can adjust capacity as your needs change.

Create a table

Use the DynamoDbClient's `createTable` method to create a new DynamoDB table. You need to construct table attributes and a table schema, both of which are used to identify the primary key of your table. You must also supply initial provisioned throughput values and a table name.

> **Note**
> If a table with the name you chose already exists, an DynamoDbException is thrown.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

Create a table with a simple primary key

This code creates a table with a simple primary key ("Name").

Code

```
    public static String createTable(DynamoDbClient ddb, String tableName, String key) {

        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
                .attributeDefinitions(AttributeDefinition.builder()
                        .attributeName(key)
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(KeySchemaElement.builder()
                        .attributeName(key)
                        .keyType(KeyType.HASH)
                        .build())
                .provisionedThroughput(ProvisionedThroughput.builder()
                        .readCapacityUnits(new Long(10))
                        .writeCapacityUnits(new Long(10))
                        .build())
                .tableName(tableName)
                .build();

        String newTable ="";
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            // Wait until the Amazon DynamoDB table is created
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
```

```
            newTable = response.tableDescription().tableName();
            return newTable;

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

See the complete example on GitHub.

Create a table with a composite primary key

Add another AttributeDefinition and KeySchemaElement to CreateTableRequest.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

Code

```
    public static String createTableComKey(DynamoDbClient ddb, String tableName) {
        CreateTableRequest request = CreateTableRequest.builder()
                .attributeDefinitions(
                        AttributeDefinition.builder()
                                .attributeName("Language")
                                .attributeType(ScalarAttributeType.S)
                                .build(),
                        AttributeDefinition.builder()
                                .attributeName("Greeting")
                                .attributeType(ScalarAttributeType.S)
                                .build())
                .keySchema(
                        KeySchemaElement.builder()
                                .attributeName("Language")
                                .keyType(KeyType.HASH)
                                .build(),
                        KeySchemaElement.builder()
                                .attributeName("Greeting")
                                .keyType(KeyType.RANGE)
                                .build())
                .provisionedThroughput(
                        ProvisionedThroughput.builder()
                                .readCapacityUnits(new Long(10))
                                .writeCapacityUnits(new Long(10)).build())
                .tableName(tableName)
                .build();

       String tableId = "";

       try {
            CreateTableResponse result = ddb.createTable(request);
```

```
            tableId = result.tableDescription().tableId();
            return tableId;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
       return "";
    }
```

See the complete example on GitHub.

List tables

You can list the tables in a particular region by calling the DynamoDbClient's `listTables` method.

> **Note**
> If the named table doesn't exist for your account and region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.List;
```

Code

```
    public static void listAllTables(DynamoDbClient ddb){

        boolean moreTables = true;
        String lastName = null;

        while(moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();

                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }
            } catch (DynamoDbException e) {
```

```
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
```

By default, up to 100 tables are returned per call—use `lastEvaluatedTableName` on the returned ListTablesResponse object to get the last table that was evaluated. You can use this value to start the listing after the last returned value of the previous listing.

See the complete example on GitHub.

Describe (get information about) a table

Call the DynamoDbClient's `describeTable` method.

> **Note**
> If the named table doesn't exist for your account and region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;
```

Code

```
    public static void describeDymamoDBTable(DynamoDbClient ddb,String tableName ) {

        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo =
                    ddb.describeTable(request).table();

            if (tableInfo != null) {
                System.out.format("Table name  : %s\n",
                        tableInfo.tableName());
                System.out.format("Table ARN   : %s\n",
                        tableInfo.tableArn());
                System.out.format("Status      : %s\n",
                        tableInfo.tableStatus());
                System.out.format("Item count  : %d\n",
                        tableInfo.itemCount().longValue());
                System.out.format("Size (bytes): %d\n",
                        tableInfo.tableSizeBytes().longValue());

                ProvisionedThroughputDescription throughputInfo =
                        tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
                        throughputInfo.readCapacityUnits().longValue());
                System.out.format("  Write Capacity: %d\n",
```

```
                        throughputInfo.writeCapacityUnits().longValue());

                List<AttributeDefinition> attributes =
                        tableInfo.attributeDefinitions();
                System.out.println("Attributes");

                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n",
                            a.attributeName(), a.attributeType());
                }
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
```

See the complete example on GitHub.

Modify (update) a table

You can modify your table's provisioned throughput values at any time by calling the DynamoDbClient's `updateTable` method.

> **Note**
> If the named table doesn't exist for your account and region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
    public static void updateDynamoDBTable(DynamoDbClient ddb,
                                           String tableName,
                                           Long readCapacity,
                                           Long writeCapacity) {

        System.out.format(
                "Updating %s with new provisioned throughput values\n",
                tableName);
        System.out.format("Read capacity : %d\n", readCapacity);
        System.out.format("Write capacity : %d\n", writeCapacity);

        ProvisionedThroughput tableThroughput = ProvisionedThroughput.builder()
                .readCapacityUnits(readCapacity)
                .writeCapacityUnits(writeCapacity)
                .build();

        UpdateTableRequest request = UpdateTableRequest.builder()
                .provisionedThroughput(tableThroughput)
                .tableName(tableName)
                .build();

        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }
```

See the complete example on GitHub.

Delete a table

Call the DynamoDbClient's `deleteTable` method and pass it the table's name.

> **Note**
> If the named table doesn't exist for your account and region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

Code

```
    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName +" was successfully deleted!");
    }
```

See the complete example on GitHub.

More information

- Guidelines for Working with Tables in the Amazon DynamoDB Developer Guide
- Working with Tables in DynamoDB in the Amazon DynamoDB Developer Guide

Working with items in DynamoDB

In DynamoDB, an item is a collection of *attributes*, each of which has a *name* and a *value*. An attribute value can be a scalar, set, or document type. For more information, see Naming Rules and Data Types in the Amazon DynamoDB Developer Guide.

Retrieve (get) an item from a table

Call the DynamoDbClient's `getItem` method and pass it a GetItemRequest object with the table name and primary key value of the item you want. It returns a GetItemResponse object with all of the

attributes for that item. You can specify one or more projection expressions in the `GetItemRequest` to retrieve specific attributes.

You can use the returned `GetItemResponse` object's `item()` method to retrieve a Map of key (String) and value (AttributeValue) pairs that are associated with the item.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
```

Code

```
    public static void getDynamoDBItem(DynamoDbClient ddb,String tableName,String
 key,String keyVal ) {

        HashMap<String,AttributeValue> keyToGet = new HashMap<String,AttributeValue>();

        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal).build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName(tableName)
                .build();

        try {
            Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1, returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", key);
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Retrieve (get) an item from a table using the asynchronous client

Invoke the `getItem` method of the DynamoDbAsyncClient and pass it a GetItemRequest object with the table name and primary key value of the item you want.

You can return a Collection instance with all of the attributes for that item (refer to the following example).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
    public static void getItem(DynamoDbAsyncClient client, String tableName, String key,
 String keyVal) {

        HashMap<String, AttributeValue> keyToGet =
                new HashMap<String, AttributeValue>();

        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal).build());

        try {

            // Create a GetItemRequest instance
            GetItemRequest request = GetItemRequest.builder()
                    .key(keyToGet)
                    .tableName(tableName)
                    .build();

            // Invoke the DynamoDbAsyncClient object's getItem
            java.util.Collection<AttributeValue> returnedItem =
 client.getItem(request).join().item().values();

            // Convert Set to Map
            Map<String, AttributeValue> map =
 returnedItem.stream().collect(Collectors.toMap(AttributeValue::s, s->s));
            Set<String> keys = map.keySet();
            for (String sinKey : keys) {
                System.out.format("%s: %s\n", sinKey, map.get(sinKey).toString());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

See the complete example on GitHub.

Add a new item to a table

Create a Map of key-value pairs that represent the item's attributes. These must include values for the table's primary key fields. If the item identified by the primary key already exists, its fields are *updated* by the request.

> **Note**
> If the named table doesn't exist for your account and region, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;
```

Code

```
    public static void putItemInTable(DynamoDbClient ddb,
                                      String tableName,
                                      String key,
                                      String keyVal,
                                      String albumTitle,
                                      String albumTitleValue,
                                      String awards,
                                      String awardVal,
                                      String songTitle,
                                      String songTitleVal){

        HashMap<String,AttributeValue> itemValues = new HashMap<String,AttributeValue>();

        // Add all content to the table
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle, AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();

        try {
            ddb.putItem(request);
            System.out.println(tableName +" was successfully updated");

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be found.\n",
tableName);
            System.err.println("Be sure that it exists and that you've typed its name
correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Update an existing item in a table

You can update an attribute for an item that already exists in a table by using the DynamoDbClient's `updateItem` method, providing a table name, primary key value, and a map of fields to update.

> **Note**
> If the named table doesn't exist for your account and region, or if the item identified by the primary key you passed in doesn't exist, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```

Code

```
    public static void updateTableItem(DynamoDbClient ddb,
                                       String tableName,
                                       String key,
                                       String keyVal,
                                       String name,
                                       String updateVal){

        HashMap<String,AttributeValue> itemKey = new HashMap<String,AttributeValue>();

        itemKey.put(key, AttributeValue.builder().s(keyVal).build());

        HashMap<String,AttributeValueUpdate> updatedValues =
                new HashMap<String,AttributeValueUpdate>();

        // Update the column specified by name with updatedVal
        updatedValues.put(name, AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s(updateVal).build())
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }
```

See the complete example on GitHub.

Delete an existing item in a table

You can delete an item that exists in a table by using the DynamoDbClient's `deleteItem` method and providing a table name as well as the primary key value.

> **Note**
> If the named table doesn't exist for your account and region, or if the item identified by the primary key you passed in doesn't exist, a ResourceNotFoundException is thrown.

Imports

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;
```

Code

```
 public static void deleteDymamoDBItem(DynamoDbClient ddb, String tableName, String key,
String keyVal) {

       HashMap<String,AttributeValue> keyToGet =
               new HashMap<String,AttributeValue>();

       keyToGet.put(key, AttributeValue.builder()
               .s(keyVal)
               .build());

       DeleteItemRequest deleteReq = DeleteItemRequest.builder()
               .tableName(tableName)
               .key(keyToGet)
               .build();

       try {
           ddb.deleteItem(deleteReq);
       } catch (DynamoDbException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

See the complete example on GitHub.

More information

- Guidelines for Working with Items in the Amazon DynamoDB Developer Guide
- Working with Items in DynamoDB in the Amazon DynamoDB Developer Guide

Mapping items in DynamoDB tables

The DynamoDB enhanced client is a high-level library that is part of the AWS SDK for Java 2.x. It offers a straightforward way to map client-side classes to DynamoDB tables. You define the relationships between tables and their corresponding model classes in your code. After you define those relationships, you can intuitively perform various create, read, update, or delete (CRUD) operations on tables or items in DynamoDB.

To begin working with the enhanced client in your project, add a dependency on the Maven artifact `dynamodb-enhanced` in addition to the `dynamodb` artifact to your build file, as shown in one of the following examples.

Maven

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.18.10</version>
        <type>pom</type>
```

```
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
  </dependencies>
  ...
</project>
```

Perform a search of the Maven central repository for the latest version.

Gradle

```
repositories {
    mavenCentral()
}
dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.18.10"))
    implementation("software.amazon.awssdk:dynamodb")
    implementation("software.amazon.awssdk:dynamodb-enhanced")
    ...
}
```

Perform a search of the Maven central repository for the latest version.

Create a `DynamoDbEnhancedClient`

The `DynamoDbEnhancedClient` instance is used to work with DynamoDB tables and mapped classes. You create a `DynamoDbEnhancedClient` from an existing `DynamoDbClient` object, as shown in the following example.

```
DynamoDbClient ddb = DynamoDbClient.builder()
        .region(Region.US_EAST_1)
        .build();

DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

// Use the enhancedClient.
```

Generate a `TableSchema`

When you work with the mapping features of the enhanced client, the first step you take is to generate a `TableSchema`. You can do this in a couple of ways.

Use an annotated Java bean

The SDK for Java 2.x includes a set of annotations that you can use with a Java bean to quickly generate a `TableSchema` for mapping your classes to tables.

Start by creating a Java data class that includes a default public constructor and standardized names of getters and setters for each property in the class. Include a class-level annotation to indicate it is a

`DynamoDbBean` and, at a minimum, include a `DynamoDbPartitionKey` annotation on the getter or setter for the primary key of the table record.

The following `Customer` class shows these annotations that will link the class definition to the DynamoDB table.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.dynamodb;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbBean;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

/**
 * This class is used by the Enhanced Client examples.
 */

 @DynamoDbBean
 public class Customer {

        private String id;
        private String name;
        private String email;
        private Instant regDate;

        @DynamoDbPartitionKey
        public String getId() {
            return this.id;
        }

        public void setId(String id) {

            this.id = id;
        }

        public String getCustName() {
            return this.name;
        }

        public void setCustName(String name) {
            this.name = name;
        }

        @DynamoDbSortKey
        public String getEmail() {
            return this.email;
        }

        public void setEmail(String email) {
            this.email = email;
        }

        public Instant getRegistrationDate() {
            return regDate;
        }
        public void setRegistrationDate(Instant registrationDate) {
            this.regDate = registrationDate;
        }
```

```
        @Override
        public String toString() {
            return "Customer [id=" + id + ", name=" + name + ", email=" + email
                    + ", regDate=" + regDate + "]";
        }
    }
```

Once you have created an annotated Java bean, use it to create the `TableSchema`, as shown in the following snippet.

```
TableSchema<Customer> customerTableSchema = TableSchema.fromBean(Customer.class);
```

Use a builder

If you would prefer to skip the slightly costly bean introspection for a faster solution, you can instead declare your schema directly and let the compiler do the heavy lifting. If you do it this way, your class does not need to follow bean naming standards nor does it need to be annotated. This following example uses a builder and is equivalent to the bean example.

```
TableSchema<Customer> customerTableSchema =
  TableSchema.builder(Customer.class)
    .newItemSupplier(Customer::new)
    .addAttribute(String.class, a -> a.name("id")
                                      .getter(Customer::getId)
                                      .setter(Customer::setId)
                                      .tags(primaryPartitionKey()))
    .addAttribute(Integer.class, a -> a.name("email")
                                       .getter(Customer::getEmail)
                                       .setter(Customer::setEmail)
                                       .tags(primarySortKey()))
    .addAttribute(String.class, a -> a.name("name")
                                      .getter(Customer::getCustName)
                                      .setter(Customer::setCustName))
    .addAttribute(Instant.class, a -> a.name("registrationDate")
                                       .getter(Customer::getRegistrationDate)
                                       .setter(Customer::setRegistrationDate))
    .build();
```

Create a table using the enhanced client

The following example shows you how to create a DynamoDB table from the `Customer` Java data bean class.

To create the `DynamoDbTable` object, pass the table name and the `TableSchema` to the `table()` method of the enhanced client. This example creates a table with the name `Customer`—identical to the class name—but the table name can be something else. Whatever you name the table, you must use this name in additional applications to work with the table. Supply this name to the `table()` method anytime you create another `DynamoDbTable` object.

The lambda parameter, `builder`, passed to the `createTable` method allows you to customize the table. The example also uses a `DynamoDbWaiter`. Since the creation of a table takes a bit of time, using a waiter avoids the need for you to write logic that polls the DynamoDB service to see if the table exists before using the table.

Imports

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
```

```
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

Code

```
public class EnhancedCreateTable {
    public static void createTable(DynamoDbEnhancedClient enhancedClient) {
        // Create a DynamoDbTable object
        DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));
        // Create the table
        customerTable.createTable(builder -> builder
                .provisionedThroughput(b -> b
                        .readCapacityUnits(10L)
                        .writeCapacityUnits(10L)
                        .build())
        );

        System.out.println("Waiting for table creation...");

        try (DynamoDbWaiter waiter = DynamoDbWaiter.create()) { // DynamoDbWaiter is
 Autocloseable
            ResponseOrException<DescribeTableResponse> response = waiter
                    .waitUntilTableExists(builder -> builder.tableName("Customer").build())
                    .matched();
            DescribeTableResponse tableDescription = response.response().orElseThrow(
                    () -> new RuntimeException("Customer table was not created."));
            // The actual error can be inspected in response.exception()
            System.out.println(tableDescription.table().tableName() + " was created.");
        }
    }
}
```

See the complete example on GitHub.

Retrieve (get) an item from a table

To get an item from a DynamoDB table, create a DynamoDbTable object and call `getItem()` with a GetItemEnhancedRequest object. As an alternative to passing in a `GetItemEnhancedRequest` object, you can take advantage of lambda expressions and the SDK's builder pattern. In the example below, the `getItem()` method takes a `Consumer<GetItemEnhancedRequest.Builder>` that ultimately creates the `GetItemEnhancedRequest` object for you.

For explanatory purposes, the example uses full lambda syntax, `(GetItemEnhancedRequest.Builder requestBuilder) -> requestBuilder.key(key)`, but a simpler expression such as `rb # rb.key(key)` works as well.

The following code snippet demonstrates the use of the enhanced client to get information from an item in a DynamoDB table.

Imports

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
```

```
import software.amazon.awssdk.enhanced.dynamodb.model.GetItemEnhancedRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

Code

```
    public static String getItem(DynamoDbEnhancedClient enhancedClient) {

        Customer result = null;

        try {
            DynamoDbTable<Customer> table = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));
            Key key = Key.builder()
                .partitionValue("id101").sortValue("tred@noserver.com")
                .build();

            // Get the item by using the key.
            result = table.getItem(
                    (GetItemEnhancedRequest.Builder requestBuilder) ->
 requestBuilder.key(key));
            System.out.println("******* The description value is " + result.getCustName());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return result.getCustName();
    }
```

See the complete example on GitHub.

Add a new item to a table

To insert a new item into a table using the enhanced client, you create an instance of the Java bean data class that is annotated with `@DynamoDbBean`. Use the setters of the Java bean instance to add the data that you want to insert. Use the `putItem()` method of the `DynamoDbTable` to perform the insertion.

The following example shows one item added to the `Customer` table.

Imports

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
```

Code

```
    public static void putRecord(DynamoDbEnhancedClient enhancedClient) {
        try {
```

```
            DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));

            // Create an Instant value.
            LocalDate localDate = LocalDate.parse("2020-04-07");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            // Populate the Table.
            Customer custRecord = new Customer();
            custRecord.setCustName("Tom red");
            custRecord.setId("id101");
            custRecord.setEmail("tred@noserver.com");
            custRecord.setRegistrationDate(instant) ;

            // Put the customer data into an Amazon DynamoDB table.
            custTable.putItem(custRecord);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Customer data added to the table with id id101");
    }
```

See the complete example on GitHub.

Batch create (put) and delete items

You can batch a series of put requests (PutItemEnhancedRequest) and delete requests (DeleteItemEnhancedRequest) to one or more tables, and then send all of the changes in a single request.

The SDK offers the builder pattern to create the `PutItemEnhancedRequest` for you. A lambda expression, such as `builder -> builder.item(record2)` is all you need to provide to the `addPutItem()` method as shown in the example below.

A DynamoDbTable object is created and three items are queued up to be added to the Customer table in the first call to `WriteBatch.builder`. You can call `addDeleteItem()` and `addPutItem()` (part of WriteBatch.Builder) multiple times in each batch, as needed.

To queue up changes to a different table, make another call to `WriteBatch.builder()` and provide a corresponding DynamoDbTable object in `mappedTableResource()`. This is shown below in the second `WriteBatch.builder()` call using the Music table and deleting one item from the table.

Imports

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;

import java.time.Instant;
import java.time.LocalDate;
```

```
import java.time.LocalDateTime;
import java.time.ZoneOffset;
```

Code

```
    public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {

        try {
            DynamoDbTable<Customer> customerMappedTable = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));
            DynamoDbTable<Music> musicMappedTable = enhancedClient.table("Music",
 TableSchema.fromBean(Music.class));
            LocalDate localDate = LocalDate.parse("2020-04-07");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            Customer record2 = new Customer();
            record2.setCustName("Fred Pink");
            record2.setId("id110");
            record2.setEmail("fredp@noserver.com");
            record2.setRegistrationDate(instant) ;

            Customer record3 = new Customer();
            record3.setCustName("Susan Pink");
            record3.setId("id120");
            record3.setEmail("spink@noserver.com");
            record3.setRegistrationDate(instant) ;

            Customer record4 = new Customer();
            record4.setCustName("Jerry orange");
            record4.setId("id101");
            record4.setEmail("jorange@noserver.com");
            record4.setRegistrationDate(instant) ;


            BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest =
 BatchWriteItemEnhancedRequest.builder()
                    .writeBatches(
                            WriteBatch.builder(Customer.class)         // add items to the
 Customer table
                                    .mappedTableResource(customerMappedTable)
                                    .addPutItem(builder -> builder.item(record2))
                                    .addPutItem(builder -> builder.item(record3))
                                    .addPutItem(builder -> builder.item(record4))
                                    .build(),
                            WriteBatch.builder(Music.class)            // delete an item
 from the Music table
                                    .mappedTableResource(musicMappedTable)
                                    .addDeleteItem(builder -> builder.key(
                                            Key.builder().partitionValue("Famous
 Band").build()))
                                    .build())
                    .build();


            // Add three items to the Customer table and delete one item from the Music
 table
            enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);

            System.out.println("done");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
```

See the complete example on GitHub.

Use a filtered query to get items from a table

You can get items from a table based on filterable queries, and then perform operations on the query results.

For the example below, assume the `Customer` table contains the following items:

id	email	custName	registrationDate
id120	spink@noserver.com	Susan Pink	2020-04-07T00:00:00Z
id101	jorange@noserver.com	Jerry orange	2020-04-07T00:00:00Z
id101	tred@noserver.com	Tom red	2020-04-07T00:00:00Z
ed110	fredp@noserver.com	Fred Pink	2020-04-07T00:00:00Z

The following steps describes what is happening in the `queryTableFilter` method below.

1. Build an expression to filter the query.
   a. You build a filter by first defining the value to match on as an AttributeValue object ("Tom red" in this example).
   b. You create a `HashMap` to hold a token as the map's key (":value" in this example) and the `AttributeValue` object as the map's value.
   c. You then build an `Expression` using a filter expression ("custName = :value") and the expression values in the map.
2. Build a `QueryConditional` object.
   - You build a `QueryConditional` object to select items based on the primary key value "id101".
3. Build the query request and execute the query.
   - Pass a lambda parameter to the DynamoDbTable's `query()` method. The SDK uses the parameter's logic to build the final query request object to send to the DynamoDB service.
4. Process the results.
   - This example processes the iterable results returned from the query in a *for*-each loop by counting the number of items returned and printing out the `Customer` object.

The following log output shows the request that is sent to the DynamoDB service.

```
DEBUG org.apache.http.wire:87 - http-outgoing-0 >>
 "{"TableName":"Customer","FilterExpression":"custName
 = :value","KeyConditionExpression":"#AMZN_MAPPED_id
 = :AMZN_MAPPED_id","ExpressionAttributeNames":
{"#AMZN_MAPPED_id":"id"},"ExpressionAttributeValues":{":AMZN_MAPPED_id":
{"S":"id101"},":value":{"S":"Tom red"}}}"
```

Note
The QueryConditional interface has several methods you can use to build your queries, including common conditional statements like greater than, less than, and in between.

Imports

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Expression;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.QueryConditional;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;

import java.util.HashMap;
import java.util.Map;
```

Code

```
    public static Integer queryTableFilter(DynamoDbEnhancedClient enhancedClient) {

        Integer countOfCustomers = 0;

        try {
            DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));

            AttributeValue att = AttributeValue.builder()
                    .s("Tom red")
                    .build();

            Map<String, AttributeValue> expressionValues = new HashMap<>();
            expressionValues.put(":value", att);

            Expression expression = Expression.builder()
                    .expression("custName = :value")
                    .expressionValues(expressionValues)
                    .build();

            // Create a QueryConditional object to query by partitionValue.
            // Since the Customer table has a sort key attribute (email), we can use an
 expression
            // to filter the query results if multiple items have the same partition key
 value.
            QueryConditional queryConditional = QueryConditional
                    .keyEqualTo(Key.builder().partitionValue("id101")
                            .build());

            // Perform the query

            for (Customer customer : mappedTable.query(
                    r -> r.queryConditional(queryConditional)
                            .filterExpression(expression)
            ).items()) {
                countOfCustomers++;
                System.out.println(customer);
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
        System.out.println("Done");
        return countOfCustomers;
```

The code returns one item. The `QueryConditional` object matches items that have an id of "id101" and the filter expression further restricts the items that are returned to items where `custName` is equal to "Tom red".

See the complete example on GitHub.

Retrieve (get) all items from a table

When you want to get all of the records in a given DynamoDB table, use the `scan()` method of your DynamoDbTable object and the `items()` method to get access to each item, against which you can execute various operations. For example, the following code snippet prints out the id and customer name of each item in the `Customer` table.

Imports

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.Iterator;
```

Code

```
    public static void scan( DynamoDbEnhancedClient enhancedClient) {
        try{
            DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));
            Iterator<Customer> results = custTable.scan().items().iterator();
            while (results.hasNext()) {
                Customer rec = results.next();
                System.out.println("The record id is "+rec.getId());
                System.out.println("The name is " +rec.getCustName());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

See the complete example on GitHub.

For more information, see Working with items in DynamoDB in the Amazon DynamoDB Developer Guide.

Working with Amazon EC2

This section provides examples of programming Amazon EC2 that use the AWS SDK for Java 2.x.

Topics

- Manage Amazon EC2 instances (p. 142)

- Use elastic IP addresses in Amazon EC2 (p. 146)
- Use regions and availability zones (p. 149)
- Work with Amazon EC2 key pairs (p. 151)
- Work with security groups in Amazon EC2 (p. 153)
- Work with Amazon EC2 instance metadata (p. 156)

Manage Amazon EC2 instances

Create an instance

Create a new Amazon EC2 instance by calling the Ec2Client's `runInstances` method, providing it with a RunInstancesRequest containing the Amazon Machine Image (AMI) to use and an instance type.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
   public static String createEC2Instance(Ec2Client ec2,String name, String amiId ) {

        RunInstancesRequest runRequest = RunInstancesRequest.builder()
                .imageId(amiId)
                .instanceType(InstanceType.T1_MICRO)
                .maxCount(1)
                .minCount(1)
                .build();

        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceId = response.instances().get(0).instanceId();

        Tag tag = Tag.builder()
                .key("Name")
                .value(name)
                .build();

        CreateTagsRequest tagRequest = CreateTagsRequest.builder()
                .resources(instanceId)
                .tags(tag)
                .build();

        try {
            ec2.createTags(tagRequest);
            System.out.printf(
                    "Successfully started EC2 Instance %s based on AMI %s",
                    instanceId, amiId);

          return instanceId;

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }

        return "";
    }
```

See the complete example on GitHub.

Start an instance

To start an Amazon EC2 instance, call the Ec2Client's `startInstances` method, providing it with a StartInstancesRequest containing the ID of the instance to start.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

Code

```
    public static void startInstance(Ec2Client ec2, String instanceId) {

        StartInstancesRequest request = StartInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        ec2.startInstances(request);
        System.out.printf("Successfully started instance %s", instanceId);
    }
```

See the complete example on GitHub.

Stop an instance

To stop an Amazon EC2 instance, call the Ec2Client's `stopInstances` method, providing it with a StopInstancesRequest containing the ID of the instance to stop.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
```

Code

```
    public static void stopInstance(Ec2Client ec2, String instanceId) {

        StopInstancesRequest request = StopInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        ec2.stopInstances(request);
        System.out.printf("Successfully stopped instance %s", instanceId);
    }
```

See the complete example on GitHub.

Reboot an instance

To reboot an Amazon EC2 instance, call the Ec2Client's `rebootInstances` method, providing it with a RebootInstancesRequest containing the ID of the instance to reboot.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.RebootInstancesRequest;
```

Code

```
    public static void rebootEC2Instance(Ec2Client ec2, String instanceId) {

      try {
            RebootInstancesRequest request = RebootInstancesRequest.builder()
                .instanceIds(instanceId)
                    .build();

            ec2.rebootInstances(request);
            System.out.printf(
                "Successfully rebooted instance %s", instanceId);
    } catch (Ec2Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
     }
  }
```

See the complete example on GitHub.

Describe instances

To list your instances, create a DescribeInstancesRequest and call the Ec2Client's `describeInstances` method. It will return a DescribeInstancesResponse object that you can use to list the Amazon EC2 instances for your account and region.

Instances are grouped by *reservation*. Each reservation corresponds to the call to `startInstances` that launched the instance. To list your instances, you must first call the `DescribeInstancesResponse` class' `reservations` method, and then call `instances` on each returned Reservation object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void describeEC2Instances( Ec2Client ec2){

        boolean done = false;
        String nextToken = null;

        try {
```

```
            do {
                DescribeInstancesRequest request =
 DescribeInstancesRequest.builder().maxResults(6).nextToken(nextToken).build();
                DescribeInstancesResponse response = ec2.describeInstances(request);

                for (Reservation reservation : response.reservations()) {
                    for (Instance instance : reservation.instances()) {
                        System.out.println("Instance Id is " + instance.instanceId());
                        System.out.println("Image id is "+  instance.imageId());
                        System.out.println("Instance type is "+  instance.instanceType());
                        System.out.println("Instance state name is "+
 instance.state().name());
                        System.out.println("monitoring information is "+
 instance.monitoring().state());

                }
            }
                nextToken = response.nextToken();
            } while (nextToken != null);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Results are paged; you can get further results by passing the value returned from the result object's `nextToken` method to a new request object's `nextToken` method, then using the new request object in your next call to `describeInstances`.

See the complete example on GitHub.

Monitor an instance

You can monitor various aspects of your Amazon EC2 instances, such as CPU and network utilization, available memory, and disk space remaining. To learn more about instance monitoring, see Monitoring Amazon EC2 in the Amazon EC2 User Guide for Linux Instances.

To start monitoring an instance, you must create a MonitorInstancesRequest with the ID of the instance to monitor, and pass it to the Ec2Client's `monitorInstances` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
    public static void monitorInstance( Ec2Client ec2, String instanceId) {

        MonitorInstancesRequest request = MonitorInstancesRequest.builder()
                .instanceIds(instanceId).build();

        ec2.monitorInstances(request);
        System.out.printf(
                "Successfully enabled monitoring for instance %s",
                instanceId);
    }
```

See the complete example on GitHub.

Stop instance monitoring

To stop monitoring an instance, create an UnmonitorInstancesRequest with the ID of the instance to stop monitoring, and pass it to the Ec2Client's `unmonitorInstances` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.MonitorInstancesRequest;
import software.amazon.awssdk.services.ec2.model.UnmonitorInstancesRequest;
```

Code

```
    public static void unmonitorInstance(Ec2Client ec2, String instanceId) {
        UnmonitorInstancesRequest request = UnmonitorInstancesRequest.builder()
                .instanceIds(instanceId).build();

        ec2.unmonitorInstances(request);

        System.out.printf(
                "Successfully disabled monitoring for instance %s",
                instanceId);
    }
```

See the complete example on GitHub.

More information

- RunInstances in the Amazon EC2 API Reference
- DescribeInstances in the Amazon EC2 API Reference
- StartInstances in the Amazon EC2 API Reference
- StopInstances in the Amazon EC2 API Reference
- RebootInstances in the Amazon EC2 API Reference
- MonitorInstances in the Amazon EC2 API Reference
- UnmonitorInstances in the Amazon EC2 API Reference

Use elastic IP addresses in Amazon EC2

EC2-Classic is retiring

> **Warning**
> We are retiring EC2-Classic on August 15, 2022. We recommend that you migrate from EC2-Classic to a VPC. For more information, see **Migrate from EC2-Classic to a VPC** in the Amazon EC2 User Guide for Linux Instances or the Amazon EC2 User Guide for Windows Instances. Also see the blog post EC2-Classic-Classic Networking is Retiring – Here's How to Prepare.

Allocate an elastic IP address

To use an Elastic IP address, you first allocate one to your account, and then associate it with your instance or a network interface.

To allocate an Elastic IP address, call the Ec2Client's `allocateAddress` method with an AllocateAddressRequest object containing the network type (classic Amazon EC2 or VPC).

The returned AllocateAddressResponse contains an allocation ID that you can use to associate the address with an instance, by passing the allocation ID and instance ID in a AssociateAddressRequest to the Ec2Client's `associateAddress` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static String getAllocateAddress( Ec2Client ec2, String instanceId) {

       try {
           AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
                .domain(DomainType.VPC)
                .build();

           AllocateAddressResponse allocateResponse =
                ec2.allocateAddress(allocateRequest);

           String allocationId = allocateResponse.allocationId();

           AssociateAddressRequest associateRequest =
                AssociateAddressRequest.builder()
                        .instanceId(instanceId)
                        .allocationId(allocationId)
                        .build();

            AssociateAddressResponse associateResponse =
 ec2.associateAddress(associateRequest);
            return associateResponse.associationId();

        } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
    }
```

See the complete example on GitHub.

Describe elastic IP addresses

To list the Elastic IP addresses assigned to your account, call the Ec2Client's `describeAddresses` method. It returns a DescribeAddressesResponse which you can use to get a list of Address objects that represent the Elastic IP addresses on your account.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Address;
import software.amazon.awssdk.services.ec2.model.DescribeAddressesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void describeEC2Address(Ec2Client ec2 ) {

        try {
            DescribeAddressesResponse response = ec2.describeAddresses();

            for(Address address : response.addresses()) {
                System.out.printf(
                    "Found address with public IP %s, " +
                            "domain %s, " +
                            "allocation id %s " +
                            "and NIC id %s",
                    address.publicIp(),
                    address.domain(),
                    address.allocationId(),
                    address.networkInterfaceId());
            }
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Release an elastic IP address

To release an Elastic IP address, call the Ec2Client's `releaseAddress` method, passing it a ReleaseAddressRequest containing the allocation ID of the Elastic IP address you want to release.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
```

Code

```
    public static void releaseEC2Address(Ec2Client ec2,String allocId) {

        try {
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
                .allocationId(allocId).build();

            ReleaseAddressResponse response = ec2.releaseAddress(request);

         System.out.printf(
                "Successfully released elastic IP address %s", allocId);
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
     }
```

After you release an Elastic IP address, it is released to the AWS IP address pool and might be unavailable to you afterward. Be sure to update your DNS records and any servers or devices that communicate with the address.

If you are using *EC2-Classic* or a *default VPC*, then releasing an Elastic IP address automatically disassociates it from any instance that it's associated with. To disassociate an Elastic IP address without releasing it, use the Ec2Client's `disassociateAddress` method.

If you are using a non-default VPC, you *must* use `disassociateAddress` to disassociate the Elastic IP address before you try to release it. Otherwise, Amazon EC2 returns an error (*InvalidIPAddress.InUse*).

See the complete example on GitHub.

More information

- Elastic IP Addresses in the Amazon EC2 User Guide for Linux Instances
- AllocateAddress in the Amazon EC2 API Reference
- DescribeAddresses in the Amazon EC2 API Reference
- ReleaseAddress in the Amazon EC2 API Reference

Use regions and availability zones

Describe regions

To list the Regions available to your account, call the Ec2Client's `describeRegions` method. It returns a DescribeRegionsResponse. Call the returned object's `regions` method to get a list of Region objects that represent each Region.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.Region;
import software.amazon.awssdk.services.ec2.model.AvailabilityZone;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
```

Code

```
        try {

            DescribeRegionsResponse regionsResponse = ec2.describeRegions();
            for(Region region : regionsResponse.regions()) {
                System.out.printf(
                        "Found Region %s " +
                                "with endpoint %s",
                        region.regionName(),
                        region.endpoint());
                System.out.println();
```

See the complete example on GitHub.

Describe availability zones

To list each Availability Zone available to your account, call the Ec2Client's `describeAvailabilityZones` method. It returns a DescribeAvailabilityZonesResponse. Call its `availabilityZones` method to get a list of AvailabilityZone objects that represent each Availability Zone.

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.Region;
import software.amazon.awssdk.services.ec2.model.AvailabilityZone;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
```

Code

Create the Ec2Client.

```
        software.amazon.awssdk.regions.Region region =
 software.amazon.awssdk.regions.Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
```

Then call describeAvailabilityZones() and retrieve results.

```
            DescribeAvailabilityZonesResponse zonesResponse =
                    ec2.describeAvailabilityZones();

            for(AvailabilityZone zone : zonesResponse.availabilityZones()) {
                System.out.printf(
                        "Found Availability Zone %s " +
                                "with status %s " +
                                "in region %s",
                        zone.zoneName(),
                        zone.state(),
                        zone.regionName());
                System.out.println();
```

See the complete example on GitHub.

Describe accounts

To describe your account, call the Ec2Client's `describeAccountAttributes` method. This method returns a DescribeAccountAttributesResponse object. Invoke this objects `accountAttributes` method to get a list of AccountAttribute objects. You can iterate through the list to retrieve an AccountAttribute object.

You can get your account's attribute values by invoking the AccountAttribute object's `attributeValues` method. This method returns a list of AccountAttributeValue objects. You can iterate through this second list to display the value of attributes (see the following code example).

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void describeEC2Account(Ec2Client ec2) {

        try{
```

```
            DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
            accountResults.accountAttributes().forEach(attribute -> {
                        System.out.print("\n The name of the attribute is
"+attribute.attributeName());

                        attribute.attributeValues().forEach(myValue ->
                            System.out.print("\n The value of the attribute is
"+myValue.attributeValue()));
                        }
            );

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

More information

- Regions and Availability Zones in the Amazon EC2 User Guide for Linux Instances
- DescribeRegions in the Amazon EC2 API Reference
- DescribeAvailabilityZones in the Amazon EC2 API Reference

Work with Amazon EC2 key pairs

Create a key pair

To create a key pair, call the Ec2Client's `createKeyPair` method with a CreateKeyPairRequest that contains the key's name.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void createEC2KeyPair(Ec2Client ec2,String keyName ) {

        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                .keyName(keyName).build();

            ec2.createKeyPair(request);
            System.out.printf(
                "Successfully created key pair named %s",
                keyName);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
     }
```

See the complete example on GitHub.

Describe key pairs

To list your key pairs or to get information about them, call the Ec2Client's `describeKeyPairs` method. It returns a DescribeKeyPairsResponse that you can use to access the list of key pairs by calling its `keyPairs` method, which returns a list of KeyPairInfo objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.KeyPairInfo;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void describeEC2Keys( Ec2Client ec2){

        try {
            DescribeKeyPairsResponse response = ec2.describeKeyPairs();

            for(KeyPairInfo keyPair : response.keyPairs()) {
                System.out.printf(
                    "Found key pair with name %s " +
                            "and fingerprint %s",
                    keyPair.keyName(),
                    keyPair.keyFingerprint());
             }

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
         }

    }
```

See the complete example on GitHub.

Delete a key pair

To delete a key pair, call the Ec2Client's `deleteKeyPair` method, passing it a DeleteKeyPairRequest that contains the name of the key pair to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void deleteKeys(Ec2Client ec2, String keyPair) {

       try {

           DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
```

```
                .keyName(keyPair)
                .build();

            DeleteKeyPairResponse response = ec2.deleteKeyPair(request);
             System.out.printf(
                "Successfully deleted key pair named %s", keyPair);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

More information

- Amazon EC2 Key Pairs in the Amazon EC2 User Guide for Linux Instances
- CreateKeyPair in the Amazon EC2 API Reference
- DescribeKeyPairs in the Amazon EC2 API Reference
- DeleteKeyPair in the Amazon EC2 API Reference

Work with security groups in Amazon EC2

Create a security group

To create a security group, call the Ec2Client's `createSecurityGroup` method with a CreateSecurityGroupRequest that contains the key's name.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

```
            CreateSecurityGroupRequest createRequest = CreateSecurityGroupRequest.builder()
                .groupName(groupName)
                .description(groupDesc)
                .vpcId(vpcId)
                .build();

            CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

See the complete example on GitHub.

Configure a security group

A security group can control both inbound (ingress) and outbound (egress) traffic to your Amazon EC2 instances.

To add ingress rules to your security group, use the Ec2Client's `authorizeSecurityGroupIngress` method, providing the name of the security group and the access rules (IpPermission) you want to assign to it within an AuthorizeSecurityGroupIngressRequest object. The following example shows how to add IP permissions to a security group.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.IpRange;
```

Code

First, create an Ec2Client

```
        Region region = Region.US_WEST_2;
        Ec2Client ec2 = Ec2Client.builder()
                .region(region)
                .build();
```

Then use the Ec2Client's `authorizeSecurityGroupIngress` method,

```
            IpRange ipRange = IpRange.builder()
                .cidrIp("0.0.0.0/0").build();

            IpPermission ipPerm = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(80)
                .fromPort(80)
                .ipRanges(ipRange)
                .build();

            IpPermission ipPerm2 = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(22)
                .fromPort(22)
                .ipRanges(ipRange)
                .build();

            AuthorizeSecurityGroupIngressRequest authRequest =
                AuthorizeSecurityGroupIngressRequest.builder()
                        .groupName(groupName)
                        .ipPermissions(ipPerm, ipPerm2)
                        .build();

            AuthorizeSecurityGroupIngressResponse authResponse =
            ec2.authorizeSecurityGroupIngress(authRequest);

            System.out.printf(
                "Successfully added ingress policy to Security Group %s",
                groupName);

            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

To add an egress rule to the security group, provide similar data in an AuthorizeSecurityGroupEgressRequest to the Ec2Client's `authorizeSecurityGroupEgress` method.

See the complete example on GitHub.

Describe security groups

To describe your security groups or get information about them, call the Ec2Client's `describeSecurityGroups` method. It returns a DescribeSecurityGroupsResponse that you can use to access the list of security groups by calling its `securityGroups` method, which returns a list of SecurityGroup objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
     public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {

        try {
            DescribeSecurityGroupsRequest request =
                DescribeSecurityGroupsRequest.builder()
                        .groupIds(groupId).build();

            DescribeSecurityGroupsResponse response =
                ec2.describeSecurityGroups(request);

             for(SecurityGroup group : response.securityGroups()) {
                System.out.printf(
                    "Found Security Group with id %s, " +
                            "vpc id %s " +
                            "and description %s",
                    group.groupId(),
                    group.vpcId(),
                    group.description());
            }
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Delete a security group

To delete a security group, call the Ec2Client's `deleteSecurityGroup` method, passing it a DeleteSecurityGroupRequest that contains the ID of the security group to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Code

```
    public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {

        try {
            DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
                .groupId(groupId)
                .build();

            ec2.deleteSecurityGroup(request);
            System.out.printf(
                "Successfully deleted Security Group with id %s", groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
     }
```

See the complete example on GitHub.

More information

- Amazon EC2 Security Groups in the Amazon EC2 User Guide for Linux Instances
- Authorizing Inbound Traffic for Your Linux Instances in the Amazon EC2 User Guide for Linux Instances
- CreateSecurityGroup in the Amazon EC2 API Reference
- DescribeSecurityGroups in the Amazon EC2 API Reference
- DeleteSecurityGroup in the Amazon EC2 API Reference
- AuthorizeSecurityGroupIngress in the Amazon EC2 API Reference

Work with Amazon EC2 instance metadata

A Java SDK client for the Amazon EC2 Instance Metadata Service (metadata client) allows your applications to access metadata on their local EC2 instance. The metadata client works with the local instance of IMDSv2 (Instance Metadata Service v2) and uses session-oriented requests.

Two client classes are available in the SDK. The synchronous `Ec2MetadataClient` is for blocking operations, and the `Ec2MetadataAsyncClient` is for asynchronous, non-blocking use cases.

Get started

To use the metadata client, add the `imds` Maven artifact to your project. You also need classes for an `SdkHttpClient` (or an `SdkAsyncHttpClient` for the asynchronous variant) on the classpath.

The following Maven XML shows dependency snippets for using the synchronous UrlConnectionHttpClient along with the dependency for metadata clients.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
```

```
            <version>VERSION</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>imds</artifactId>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>url-connection-client</artifactId>
    </dependency>
    <!-- other dependencies -->
</dependencies>
```

Search the Maven central repository for the latest version of the bom artifact.

To use an asynchronous HTTP client, replace the dependency snippet for the `url-connection-client` artifact. For example, the following snippet brings in the NettyNioAsyncHttpClient implementation.

```
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>netty-nio-client</artifactId>
    </dependency>
```

Use the metadata client

Instantiate a metadata client

You can instantiate an instance of a synchronous `Ec2MetadataClient` when only one implementation of the `SdkHttpClient` interface is present on the classpath. To do so, call the static `Ec2MetadataClient#create()` method as shown in the following snippet.

```
Ec2MetadataClient client = Ec2MetadataClient.create(); // 'Ec2MetadataAsyncClient#create'
 is the asynchronous version.
```

If your application has multiple implementations of the `SdkHttpClient` or `SdkHttpAsyncClient` interface, you must specify an implementation for the metadata client to use as shown in the the section called "Configurable HTTP client" (p. 159) section.

> **Note**
> For most service clients, such as Amazon S3, the SDK for Java automatically adds implementations of the `SdkHttpClient` or `SdkHttpAsyncClient` interface. If your metadata client uses the same implementation, then `Ec2MetadataClient#create()` will work. If you require a different implementation, you must specify it when you create the metadata client.

Send requests

To retrieve instance metadata, instantiate the `EC2MetadataClient` class and call the `get` method with a path parameter that specifies the instance metadata category.

The following example prints the value associated with the `ami-id` key to the console.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/ami-id");
System.out.println(response.asString());
```

```
client.close(); // Closes the internal resources used by the Ec2MetadataClient class.
```

If the path isn't valid, the `get` method throws an exception.

Reuse the same client instance for multiple requests, but call `close` on the client when it is no longer needed to release resources. After the close method is called, the client instance can't be used anymore.

Parse responses

EC2 instance metadata can be output in different formats. Plain text and JSON are the most commonly used formats. The metadata clients offer ways to work with those formats.

As the following example shows, use the `asString` method to get the data as a Java string. You can also use the `asList` method to separate a plain text response that returns multiple lines.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

If the response is in JSON, use the `Ec2MetadataResponse#asDocument` method to parse the JSON response into a Document instance as shown in the following code snippet.

```
Document fullResponse = response.asDocument();
```

An exception will be thrown if the format of the metadata is not in JSON. If the response is successfully parsed, you can use the document API to inspect the response in more detail. Consult the instance metadata category chart to learn which metadata categories deliver JSON-formatted responses.

Configure a metadata client

Retries

You can configure a metadata client with a retry mechanism. If you do, then the client can automatically retry requests that fail for unexpected reasons. By default, the client retries three times on a failed request with an exponential backoff time between attempts.

If your use case requires a different retry mechanism, you can customize the client using the `retryPolicy` method on its builder. For example, the following example shows a synchronous client configured with a fixed delay of two seconds between attempts and five retry attempts.

```
BackoffStrategy fixedBackoffStrategy =
 FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                      .retryPolicy(retryPolicyBuilder -> retryPolicyBuilder.numRetries(5)

 .backoffStrategy(fixedBackoffStrategy))
                      .build();
```

There are several BackoffStrategies that you can use with a metadata client.

You can also disable the retry mechanism entirely, as the following snippet shows.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .retryPolicy(Ec2MetadataRetryPolicy.none())
                     .build();
```

Using `Ec2MetadataRetryPolicy#none()` disables the default retry policy so that the metadata client attempts no retries.

IP version

By default, a metadata client uses the IPV4 endpoint at `http://169.254.169.254`. To change the client to use the IPV6 version, use either the `endpointMode` or the `endpoint` method of the builder. An exception results if both methods are called on the builder.

The following examples show both IPV6 options.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                      .endpointMode(EndpointMode.IPV6)
                      .build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                      .endpoint(URI.create("http://[fd00:ec2::254]"))
                      .build();
```

Key features

Asynchronous client

To use the non-blocking version of the client, instantiate an instance of the `Ec2MetadataAsyncClient` class. The code in the following example creates an asynchronous client with default settings and uses the `get` method to retrieve the value for the `ami-id` key.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/ami-
id");
```

The `java.util.concurrent.CompletableFuture` returned by the `get` method completes when the response returns. The following example prints the `ami-id` metadata to the console.

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

Configurable HTTP client

The builder for each metadata client has a `httpClient` method that you can use to supply a customized HTTP client.

The following example shows code for a custom `UrlConnectionHttpClient` instance.

```
SdkHttpClient httpClient =
    UrlConnectionHttpClient.builder()
                           .socketTimeout(Duration.ofMinutes(5))
                           .proxyConfiguration(proxy -> proxy.endpoint(URI.create("http://
proxy.example.net:8888"))))
                           .build();
Ec2MetadataClient metaDataClient =
    Ec2MetadataClient.builder()
                      .httpClient(httpClient)
                      .build();
// Use the metaDataClient instance.
metaDataClient.close();   // Close the instance when no longer needed.
```

The following example shows code for a custom `NettyNioAsyncHttpClient` instance with an asynchronous metadata client.

```
SdkAsyncHttpClient httpAsyncClient =
    NettyNioAsyncHttpClient.builder()
                            .connectionTimeout(Duration.ofMinutes(5))
                            .maxConcurrency(100)
                            .build();
Ec2MetadataAsyncClient asyncMetaDataClient =
    Ec2MetadataAsyncClient.builder()
                           .httpClient(httpAsyncClient)
                           .build();
// Use the asyncMetaDataClient instance.
asyncMetaDataClient.close();   // Close the instance when no longer needed.
```

The the section called “Configuring HTTP clients” (p. 50) topic in this guide provides details on how to configure the HTTP clients that are available in the SDK for Java.

Token caching

Because the metadata clients use IMDSv2, all requests are associated with a session. A session is defined by a token that has an expiration, which the metadata client manages for you. Every metadata request automatically reuses the token until it expires.

By default, a token lasts for six hours (21,600 seconds). We recommend that you keep the default time-to-live value, unless your specific use case requires advanced configuration.

If needed, configure the duration by using the `tokenTtl` builder method. For example, the code in the following snippet creates a client with a session duration of five minutes.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                      .tokenTtl(Duration.ofMinutes(5))
                      .build();
```

If you omit calling the `tokenTtl` method on the builder, the default duration of 21,600 is used instead.

Working with IAM

This section provides examples of programming IAM by using the AWS SDK for Java 2.x.

AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. For a complete guide to IAM, visit the IAM User Guide.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Managing IAM access keys (p. 161)
- Managing IAM Users (p. 164)
- Using IAM account aliases (p. 167)
- Working with IAM policies (p. 169)
- Working with IAM server certificates (p. 174)

Managing IAM access keys

Create an access key

To create an IAM access key, call the IamClient's `createAccessKey` method with a CreateAccessKeyRequest object.

> **Note**
> You must set the region to **AWS_GLOBAL** for IamClient calls to work because IAM is a global service.

Imports

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static String createIAMAccessKey(IamClient iam,String user) {

        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user).build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
           String keyId = response.accessKey().accessKeyId();
           return keyId;

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

See the complete example on GitHub.

List access keys

To list the access keys for a given user, create a ListAccessKeysRequest object that contains the user name to list keys for, and pass it to the IamClient's `listAccessKeys` method.

> **Note**
> If you do not supply a user name to `listAccessKeys`, it will attempt to list access keys associated with the AWS account that signed the request.

Imports

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void listKeys( IamClient iam,String userName ){

        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                        .userName(userName).build();
                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                        .userName(userName)
                        .marker(newMarker).build();
                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata :
                    response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                        metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

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

The results of `listAccessKeys` are paged (with a default maximum of 100 records per call). You can call `isTruncated` on the returned ListAccessKeysResponse object to see if the query returned fewer results then are available. If so, then call `marker` on the `ListAccessKeysResponse` and use it when creating a new request. Use that new request in the next invocation of `listAccessKeys`.

See the complete example on GitHub.

Retrieve an access key's last used time

To get the time an access key was last used, call the IamClient's `getAccessKeyLastUsed` method with the access key's ID (which can be passed in using a GetAccessKeyLastUsedRequest object.

You can then use the returned GetAccessKeyLastUsedResponse object to retrieve the key's last used time.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedRequest;
import software.amazon.awssdk.services.iam.model.GetAccessKeyLastUsedResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void getAccessKeyLastUsed(IamClient iam, String accessId ){

        try {
            GetAccessKeyLastUsedRequest request = GetAccessKeyLastUsedRequest.builder()
                    .accessKeyId(accessId).build();

            GetAccessKeyLastUsedResponse response = iam.getAccessKeyLastUsed(request);

            System.out.println("Access key was last used at: " +
                    response.accessKeyLastUsed().lastUsedDate());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

See the complete example on GitHub.

Activate or deactivate access keys

You can activate or deactivate an access key by creating an UpdateAccessKeyRequest object, providing the access key ID, optionally the user name, and the desired status, then passing the request object to the IamClient's `updateAccessKey` method.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
       public static void updateKey(IamClient iam, String username, String accessId, String
 status ) {

          try {
              if (status.toLowerCase().equalsIgnoreCase("active")) {
                  statusType = StatusType.ACTIVE;
              } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                  statusType = StatusType.INACTIVE;
              } else {
                  statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
              }
              UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                .accessKeyId(accessId)
                .userName(username)
                .status(statusType)
                .build();

              iam.updateAccessKey(request);

              System.out.printf(
                "Successfully updated the status of access key %s to" +
                        "status %s for user %s", accessId, status, username);

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

```
        }
    }
```

See the complete example on GitHub.

Delete an access key

To permanently delete an access key, call the IamClient's `deleteKey` method, providing it with a DeleteAccessKeyRequest containing the access key's ID and username.

> **Note**
> Once deleted, a key can no longer be retrieved or used. To temporarily deactivate a key so that it can be activated again later, use updateAccessKey (p. 163) method instead.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void deleteKey(IamClient iam ,String username, String accessKey ) {

        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                " from user " + username);

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

See the complete example on GitHub.

More information

- CreateAccessKey in the IAM API Reference
- ListAccessKeys in the IAM API Reference
- GetAccessKeyLastUsed in the IAM API Reference
- UpdateAccessKey in the IAM API Reference
- DeleteAccessKey in the IAM API Reference

Managing IAM Users

Creating a User

Create a new IAM user by providing the user name to the IamClient's `createUser` method using a CreateUserRequest object containing the user name.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;
```

Code

```
    public static String createIAMUser(IamClient iam, String username ) {

        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

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

See the complete example on GitHub.

Listing Users

To list the IAM users for your account, create a new ListUsersRequest and pass it to the IamClient's `listUsers` method. You can retrieve the list of users by calling `users` on the returned ListUsersResponse object.

The list of users returned by `listUsers` is paged. You can check to see there are more results to retrieve by calling the response object's `isTruncated` method. If it returns `true`, then call the response object's `marker()` method. Use the marker value to create a new request object. Then call the `listUsers` method again with the new request.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.services.iam.model.User;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void listAllUsers(IamClient iam ) {

        try {

             boolean done = false;
             String newMarker = null;

             while(!done) {
                ListUsersResponse response;

                if (newMarker == null) {
                    ListUsersRequest request = ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                        .marker(newMarker).build();
                    response = iam.listUsers(request);
                }

                for(User user : response.users()) {
                 System.out.format("\n Retrieved user %s", user.userName());
                }

                if(!response.isTruncated()) {
                  done = true;
                } else {
                    newMarker = response.marker();
                }
            }
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Updating a User

To update a user, call the IamClient object's `updateUser` method, which takes a UpdateUserRequest object that you can use to change the user's *name* or *path*.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

Code

```
    public static void updateIAMUser(IamClient iam, String curName,String newName ) {

        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
```

```
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s",
                newName);
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Deleting a User

To delete a user, call the IamClient's `deleteUser` request with a UpdateUserRequest object set with the user name to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void deleteIAMUser(IamClient iam, String userName) {

        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

More Information

- IAM Users in the IAM User Guide
- Managing IAM Users in the IAM User Guide
- CreateUser in the IAM API Reference
- ListUsers in the IAM API Reference
- UpdateUser in the IAM API Reference
- DeleteUser in the IAM API Reference

Using IAM account aliases

If you want the URL for your sign-in page to contain your company name or other friendly identifier instead of your AWS account ID, you can create an alias for your AWS account.

Note
AWS supports exactly one account alias per account.

Create an account alias

To create an account alias, call the IamClient's `createAccountAlias` method with a CreateAccountAliasRequest object that contains the alias name.

Imports

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void createIAMAccountAlias(IamClient iam, String alias) {

        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
                .accountAlias(alias)
                .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

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

See the complete example on GitHub.

List account aliases

To list your account's alias, if any, call the IamClient's `listAccountAliases` method.

Note
The returned ListAccountAliasesResponse supports the same `isTruncated` and `marker` methods as other AWS SDK for Java *list* methods, but an account can have only *one* account alias.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void listAliases(IamClient iam) {

        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();

            for (String alias : response.accountAliases()) {
```

```
                System.out.printf("Retrieved account alias %s", alias);
            }

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

see the complete example on GitHub.

Delete an account alias

To delete your account's alias, call the IamClient's `deleteAccountAlias` method. When deleting an account alias, you must supply its name using a DeleteAccountAliasRequest object.

Imports

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void deleteIAMAccountAlias(IamClient iam, String alias ) {

        try {
            DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

See the complete example on GitHub.

More information

- Your AWS Account ID and Its Alias in the IAM User Guide
- CreateAccountAlias in the IAM API Reference
- ListAccountAliases in the IAM API Reference
- DeleteAccountAlias in the IAM API Reference

Working with IAM policies

Create a policy

To create a new policy, provide the policy's name and a JSON-formatted policy document in a CreatePolicyRequest to the IamClient's `createPolicy` method.

Imports

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
```

Code

```
    public static String createIAMPolicy(IamClient iam, String policyName ) {

        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                .policyName(policyName)
                .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);
            waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

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

See the complete example on GitHub.

Get a policy

To retrieve an existing policy, call the IamClient's `getPolicy` method, providing the policy's ARN within a GetPolicyRequest object.

Imports

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void getIAMPolicy(IamClient iam, String policyArn) {
```

```
        try {

            GetPolicyRequest request = GetPolicyRequest.builder()
                .policyArn(policyArn).build();

            GetPolicyResponse response = iam.getPolicy(request);
            System.out.format("Successfully retrieved policy %s",
                response.policy().policyName());

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

See the complete example on GitHub.

Attach a role policy

You can attach a policy to an IAM role by calling the IamClient's `attachRolePolicy` method, providing it with the role name and policy ARN in an AttachRolePolicyRequest.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn ) {

        try {

             ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse  response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy: attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn)==0) {
                   System.out.println(roleName +
                            " policy is already attached to this role.");
                    return;
                }
          }

            AttachRolePolicyRequest attachRequest =
```

```
                AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policyArn)
                        .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                " to role " + roleName);

        } catch (IamException e) {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
         }

    System.out.println("Done");
   }
```

See the complete example on GitHub.

List attached role policies

List attached policies on a role by calling the IamClient's `listAttachedRolePolicies` method. It takes a ListAttachedRolePoliciesRequest object that contains the role name to list the policies for.

Call `getAttachedPolicies` on the returned ListAttachedRolePoliciesResponse object to get the list of attached policies. Results may be truncated; if the `ListAttachedRolePoliciesResponse` object's `isTruncated` method returns `true`, call the `ListAttachedRolePoliciesResponse` object's `marker` method. Use the marker returned to create a new request and use it to call `listAttachedRolePolicies` again to get the next batch of results.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;
```

Code

```
   public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn ) {

       try {

            ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                  .roleName(roleName)
                  .build();

           ListAttachedRolePoliciesResponse  response =
 iam.listAttachedRolePolicies(request);
           List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

           // Ensure that the policy is not attached to this role
           String polArn = "";
           for (AttachedPolicy policy: attachedPolicies) {
```

```
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn)==0) {
                   System.out.println(roleName +
                            " policy is already attached to this role.");
                    return;
                }
         }

           AttachRolePolicyRequest attachRequest =
                AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policyArn)
                        .build();

           iam.attachRolePolicy(attachRequest);

           System.out.println("Successfully attached policy " + policyArn +
                " to role " + roleName);

        } catch (IamException e) {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
         }

     System.out.println("Done");
    }
```

See the {https---github-com-awsdocs-aws-doc-sdk-examples-blob-master-javav2-example-code-iam-src-main-java-com-example-iam-AttachRolePolicy-java}[complete example] on GitHub.

Detach a role policy

To detach a policy from a role, call the IamClient's `detachRolePolicy` method, providing it with the role name and policy ARN in a DetachRolePolicyRequest.

Imports

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void detachPolicy(IamClient iam, String roleName, String policyArn ) {

        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                " from role " + roleName);

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

See the complete example on GitHub.

More information

- Overview of IAM Policies in the IAM User Guide.
- AWSIAM Policy Reference in the IAM User Guide.
- CreatePolicy in the IAM API Reference
- GetPolicy in the IAM API Reference
- AttachRolePolicy in the IAM API Reference
- ListAttachedRolePolicies in the IAM API Reference
- DetachRolePolicy in the IAM API Reference

Working with IAM server certificates

To enable HTTPS connections to your website or application on AWS, you need an SSL/TLS *server certificate*. You can use a server certificate provided by AWS Certificate Manager or one that you obtained from an external provider.

We recommend that you use ACM to provision, manage, and deploy your server certificates. With ACM you can request a certificate, deploy it to your AWS resources, and let ACM handle certificate renewals for you. Certificates provided by ACM are free. For more information about ACM, see the AWS Certificate Manager User Guide.

Get a server certificate

You can retrieve a server certificate by calling the IamClient's `getServerCertificate` method, passing it a GetServerCertificateRequest with the certificate's name.

Imports

```
import software.amazon.awssdk.services.iam.model.GetServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.GetServerCertificateResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void getCertificate(IamClient iam,String certName ) {

        try {
            GetServerCertificateRequest request = GetServerCertificateRequest.builder()
                    .serverCertificateName(certName)
                    .build();

            GetServerCertificateResponse response = iam.getServerCertificate(request);
            System.out.format("Successfully retrieved certificate with body %s",
                response.serverCertificate().certificateBody());

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

See the complete example on GitHub.

List server certificates

To list your server certificates, call the IamClient's `listServerCertificates` method with a ListServerCertificatesRequest. It returns a ListServerCertificatesResponse.

Call the returned `ListServerCertificateResponse` object's `serverCertificateMetadataList` method to get a list of ServerCertificateMetadata objects that you can use to get information about each certificate.

Results may be truncated; if the `ListServerCertificateResponse` object's `isTruncated` method returns `true`, call the `ListServerCertificatesResponse` object's `marker` method and use the marker to create a new request. Use the new request to call `listServerCertificates` again to get the next batch of results.

Imports

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesRequest;
import software.amazon.awssdk.services.iam.model.ListServerCertificatesResponse;
import software.amazon.awssdk.services.iam.model.ServerCertificateMetadata;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

Code

```
    public static void listCertificates(IamClient iam) {

        try {
            boolean done = false;
            String newMarker = null;

            while(!done) {
              ListServerCertificatesResponse response;

            if (newMarker == null) {
                ListServerCertificatesRequest request =
                        ListServerCertificatesRequest.builder().build();
                response = iam.listServerCertificates(request);
            } else {
                ListServerCertificatesRequest request =
                        ListServerCertificatesRequest.builder()
                                .marker(newMarker).build();
                response = iam.listServerCertificates(request);
            }

            for(ServerCertificateMetadata metadata :
                    response.serverCertificateMetadataList()) {
                System.out.printf("Retrieved server certificate %s",
                        metadata.serverCertificateName());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

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

```
    }
```

See the complete example on GitHub.

Update a server certificate

You can update a server certificate's name or path by calling the IamClient's `updateServerCertificate` method. It takes a UpdateServerCertificateRequest object set with the server certificate's current name and either a new name or new path to use.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateRequest;
import software.amazon.awssdk.services.iam.model.UpdateServerCertificateResponse;
```

Code

```
    public static void updateCertificate(IamClient iam, String curName, String newName) {

        try {
            UpdateServerCertificateRequest request =
                UpdateServerCertificateRequest.builder()
                        .serverCertificateName(curName)
                        .newServerCertificateName(newName)
                        .build();

            UpdateServerCertificateResponse response =
                iam.updateServerCertificate(request);


            System.out.printf("Successfully updated server certificate to name %s",
                newName);

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

See the complete example on GitHub.

Delete a server certificate

To delete a server certificate, call the IamClient's `deleteServerCertificate` method with a DeleteServerCertificateRequest containing the certificate's name.

Imports

```
import software.amazon.awssdk.services.iam.model.DeleteServerCertificateRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

Code

```
    public static void deleteCert(IamClient iam,String certName ) {

        try {
```

```
            DeleteServerCertificateRequest request =
                DeleteServerCertificateRequest.builder()
                        .serverCertificateName(certName)
                        .build();

            iam.deleteServerCertificate(request);
            System.out.println("Successfully deleted server certificate " +
                    certName);

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

See the complete example on GitHub.

More information

- Working with Server Certificates in the IAM User Guide
- GetServerCertificate in the IAM API Reference
- ListServerCertificates in the IAM API Reference
- UpdateServerCertificate in the IAM API Reference
- DeleteServerCertificate in the IAM API Reference
- AWS Certificate Manager User Guide

Working with Amazon Athena

Amazon Athena is a serverless, interactive query service to query data and analyze big data in Amazon S3 by using standard SQL. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with CloudWatch

This section provides examples of programming CloudWatch by using the AWS SDK for Java 2.x.

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define.

For more information about CloudWatch, see the Amazon CloudWatch User Guide.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Getting metrics from CloudWatch (p. 178)
- Publishing custom metric data to CloudWatch (p. 179)
- Working with CloudWatch alarms (p. 180)
- Using alarm actions in CloudWatch (p. 183)
- Sending events to CloudWatch (p. 184)

Getting metrics from CloudWatch

Listing metrics

To list CloudWatch metrics, create a ListMetricsRequest and call the CloudWatchClient's `listMetrics` method. You can use the `ListMetricsRequest` to filter the returned metrics by namespace, metric name, or dimensions.

> **Note**
> A list of metrics and dimensions that are posted by AWS services can be found within the Amazon CloudWatch Metrics and Dimensions Reference in the Amazon CloudWatch User Guide.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
```

Code

```
    public static void listMets( CloudWatchClient cw, String namespace) {

        boolean done = false;
        String nextToken = null;

        try {
            while(!done) {

                ListMetricsResponse response;

                if (nextToken == null) {
                   ListMetricsRequest request = ListMetricsRequest.builder()
                         .namespace(namespace)
                         .build();

                 response = cw.listMetrics(request);
                } else {
                  ListMetricsRequest request = ListMetricsRequest.builder()
                         .namespace(namespace)
                         .nextToken(nextToken)
                         .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf(
                        "Retrieved metric %s", metric.metricName());
                System.out.println();
            }

            if(response.nextToken() == null) {
                done = true;
            } else {
                nextToken = response.nextToken();
            }
        }

        } catch (CloudWatchException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

The metrics are returned in a ListMetricsResponse by calling its `getMetrics` method.

The results may be *paged*. To retrieve the next batch of results, call `nextToken` on the response object and use the token value to build a new request object. Then call the `listMetrics` method again with the new request.

See the complete example on GitHub.

More information

- ListMetrics in the Amazon CloudWatch API Reference

Publishing custom metric data to CloudWatch

A number of AWS services publish their own metrics in namespaces beginning with " AWS " You can also publish custom metric data using your own namespace (as long as it doesn't begin with " AWS ").

Publish custom metric data

To publish your own metric data, call the CloudWatchClient's `putMetricData` method with a PutMetricDataRequest. The `PutMetricDataRequest` must include the custom namespace to use for the data, and information about the data point itself in a MetricDatum object.

> **Note**
> You cannot specify a namespace that begins with " AWS ". Namespaces that begin with " AWS " are reserved for use by Amazon Web Services products.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
```

Code

```
    public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

        try {
            Dimension dimension = Dimension.builder()
                    .name("UNIQUE_PAGES")
                    .value("URLS")
                    .build();

            // Set an Instant object
            String time =
 ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
            Instant instant = Instant.parse(time);
```

```
            MetricDatum datum = MetricDatum.builder()
                .metricName("PAGES_VISITED")
                .unit(StandardUnit.NONE)
                .value(dataPoint)
                .timestamp(instant)
                .dimensions(dimension).build();

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                .namespace("SITE/TRAFFIC")
                .metricData(datum).build();

            cw.putMetricData(request);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Successfully put data point %f", dataPoint);
    }
```

See the complete example on GitHub.

More information

- Using Amazon CloudWatch Metrics in the Amazon CloudWatch User Guide.
- AWS Namespaces in the Amazon CloudWatch User Guide.
- PutMetricData in the Amazon CloudWatch API Reference.

Working with CloudWatch alarms

Create an alarm

To create an alarm based on a CloudWatch metric, call the CloudWatchClient's `putMetricAlarm` method with a PutMetricAlarmRequest filled with the alarm conditions.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
```

Code

```
    public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
 instanceId) {

        try {
            Dimension dimension = Dimension.builder()
                .name("InstanceId")
                .value(instanceId).build();

            PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
                .alarmName(alarmName)
```

```
                .comparisonOperator(
                        ComparisonOperator.GREATER_THAN_THRESHOLD)
                .evaluationPeriods(1)
                .metricName("CPUUtilization")
                .namespace("AWS/EC2")
                .period(60)
                .statistic(Statistic.AVERAGE)
                .threshold(70.0)
                .actionsEnabled(false)
                .alarmDescription(
                        "Alarm when server CPU utilization exceeds 70%")
                .unit(StandardUnit.SECONDS)
                .dimensions(dimension)
                .build();

            cw.putMetricAlarm(request);
            System.out.printf(
                    "Successfully created alarm with name %s", alarmName);

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

See the complete example on GitHub.

List alarms

To list the CloudWatch alarms that you have created, call the CloudWatchClient's `describeAlarms` method with a DescribeAlarmsRequest that you can use to set options for the result.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
```

Code

```
    public static void desCWAlarms( CloudWatchClient cw) {

        try {

            boolean done = false;
            String newToken = null;

            while(!done) {
                DescribeAlarmsResponse response;

                if (newToken == null) {
                    DescribeAlarmsRequest request =
 DescribeAlarmsRequest.builder().build();
                    response = cw.describeAlarms(request);
                } else {
                    DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
                        .nextToken(newToken)
                        .build();
                    response = cw.describeAlarms(request);
                }
```

```
                for(MetricAlarm alarm : response.metricAlarms()) {
                    System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
                }

                if(response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
```

The list of alarms can be obtained by calling `MetricAlarms` on the DescribeAlarmsResponse that is returned by `describeAlarms`.

The results may be *paged*. To retrieve the next batch of results, call `nextToken` on the response object and use the token value to build a new request object. Then call the `describeAlarms` method again with the new request.

> **Note**
> You can also retrieve alarms for a specific metric by using the CloudWatchClient's `describeAlarmsForMetric` method. Its use is similar to `describeAlarms`.

See the complete example on GitHub.

Delete alarms

To delete CloudWatch alarms, call the CloudWatchClient's `deleteAlarms` method with a DeleteAlarmsRequest containing one or more names of alarms that you want to delete.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
```

Code

```
    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.deleteAlarms(request);
            System.out.printf("Successfully deleted alarm %s", alarmName);

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

See the complete example on GitHub.

More information

- Creating Amazon CloudWatch Alarms in the Amazon CloudWatch User Guide
- PutMetricAlarm in the Amazon CloudWatch API Reference
- DescribeAlarms in the Amazon CloudWatch API Reference
- DeleteAlarms in the Amazon CloudWatch API Reference

Using alarm actions in CloudWatch

Using CloudWatch alarm actions, you can create alarms that perform actions such as automatically stopping, terminating, rebooting, or recovering Amazon EC2 instances.

> **Note**
> Alarm actions can be added to an alarm by using the PutMetricAlarmRequest's `alarmActions` method when creating an alarm (p. 180).

Enable alarm actions

To enable alarm actions for a CloudWatch alarm, call the CloudWatchClient's `enableAlarmActions` with a EnableAlarmActionsRequest containing one or more names of alarms whose actions you want to enable.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsResponse;
```

Code

```
    public static void enableActions(CloudWatchClient cw, String alarm) {

        try {
            EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
                .alarmNames(alarm).build();

           cw.enableAlarmActions(request);
            System.out.printf(
                    "Successfully enabled actions on alarm %s", alarm);

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

See the complete example on GitHub.

Disable alarm actions

To disable alarm actions for a CloudWatch alarm, call the CloudWatchClient's `disableAlarmActions` with a DisableAlarmActionsRequest containing one or more names of alarms whose actions you want to disable.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;
```

Code

```
    public static void disableActions(CloudWatchClient cw, String alarmName) {

        try {
             DisableAlarmActionsRequest request = DisableAlarmActionsRequest.builder()
                     .alarmNames(alarmName)
                     .build();

            cw.disableAlarmActions(request);
            System.out.printf(
                    "Successfully disabled actions on alarm %s", alarmName);

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

See the complete example on GitHub.

More information

- Create Alarms to Stop, Terminate, Reboot, or Recover an Instance in the Amazon CloudWatch User Guide
- PutMetricAlarm in the Amazon CloudWatch API Reference
- EnableAlarmActions in the Amazon CloudWatch API Reference
- DisableAlarmActions in the Amazon CloudWatch API Reference

Sending events to CloudWatch

CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources to Amazon EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, Amazon SNS topics, Amazon SQS queues, or built-in targets. You can match events and route them to one or more target functions or streams by using simple rules.

Add events

To add custom CloudWatch events, call the CloudWatchEventsClient's `putEvents` method with a PutEventsRequest object that contains one or more PutEventsRequestEntry objects that provide details about each event. You can specify several parameters for the entry such as the source and type of the event, resources associated with the event, and so on.

> **Note**
> You can specify a maximum of 10 events per call to `putEvents`.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
```

```
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;
```

Code

```
    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {

        try {

            final String EVENT_DETAILS =
                "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

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

See the complete example on GitHub.

Add rules

To create or update a rule, call the CloudWatchEventsClient's `putRule` method with a PutRuleRequest with the name of the rule and optional parameters such as the event pattern, IAM role to associate with the rule, and a scheduling expression that describes how often the rule is run.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

Code

```
    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
 roleArn) {

        try {
            PutRuleRequest request = PutRuleRequest.builder()
                .name(ruleName)
                .roleArn(roleArn)
                .scheduleExpression("rate(5 minutes)")
                .state(RuleState.ENABLED)
                .build();

            PutRuleResponse response = cwe.putRule(request);
```

```
            System.out.printf(
                    "Successfully created CloudWatch events rule %s with arn %s",
                    roleArn, response.ruleArn());
        } catch (
            CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Add targets

Targets are the resources that are invoked when a rule is triggered. Example targets include Amazon EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, and built-in targets.

To add a target to a rule, call the CloudWatchEventsClient's `putTargets` method with a PutTargetsRequest containing the rule to update and a list of targets to add to the rule.

Imports

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;
```

Code

```
    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
 functionArn, String targetId ) {

        try {
            Target target = Target.builder()
                .arn(functionArn)
                .id(targetId)
                .build();

            PutTargetsRequest request = PutTargetsRequest.builder()
                .targets(target)
                .rule(ruleName)
                .build();

            PutTargetsResponse response = cwe.putTargets(request);
            System.out.printf(
                "Successfully created CloudWatch events target for rule %s",
                ruleName);
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

More information

- Adding Events with PutEvents in the Amazon CloudWatch Events User Guide
- Schedule Expressions for Rules in the Amazon CloudWatch Events User Guide

- Event Types for CloudWatch Events in the Amazon CloudWatch Events User Guide
- Events and Event Patterns in the Amazon CloudWatch Events User Guide
- PutEvents in the Amazon CloudWatch Events API Reference
- PutTargets in the Amazon CloudWatch Events API Reference
- PutRule in the Amazon CloudWatch Events API Reference

Working with AWS CloudTrail

AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. ee the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon Cognito

With Amazon Cognito, you can quickly add user sign-up or sign-in capability to your web or mobile app. The examples here demonstrate some of the basic functionality of Amazon Cognito.

Create a user pool

A user pool is a directory of users that you can configure for your web or mobile app.

To create a user pool, start by building a CreateUserPoolRequest object, with the name of the user pool as the value of its `poolName()`. Call the `createUserPool()` method of your CreateUserPoolRequest, passing in the `CreateUserPoolRequest` object. You can capture the result of this request as a CreateUserPoolResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;
```

Code

```
    public static String createPool(CognitoIdentityProviderClient cognitoClient, String
 userPoolName ) {

        try {
            CreateUserPoolResponse response = cognitoClient.createUserPool(
                    CreateUserPoolRequest.builder()
                            .poolName(userPoolName)
                            .build()
            );
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e){
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

See the complete example on GitHub.

List users from a user pool

To list users from your user pools, start by building a ListUserPoolsRequest object, with the number of maximum results as the value of its `maxResults()`. Call the `listUserPools()` method of your `CognitoIdentityProviderClient`, passing in the `ListUserPoolsRequest` object. You can capture the result of this request as a ListUserPoolsResponse object, as demonstrated in the following code snippet. Create a UserPoolDescriptionType object to easily iterate over the results and pull out the attributes of each user.

Imports

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;
```

Code

```
    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient ) {

        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                        System.out.println("User pool " + userpool.name() + ", User ID " +
 userpool.id() );
                    }
            );

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

See the complete example on GitHub.

Create an identity pool

An identity pool is a container that organizes the IDs from your external identity provider, keeping a unique identifier for each user. To create an identity pool, start by building a CreateIdentityPoolRequest with the name of the user pool as the value of its `identityPoolName()`. Set `allowUnauthenticatedIdentities()` to `true` or `false`. Call the `createIdentityPool()` method of your `CognitoIdentityClient` object, passing in the `CreateIdentityPoolRequest` object. You can capture the result of this request as a CreateIdentityPoolResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
```

Code

```
    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName ) {

        try {
            CreateIdentityPoolRequest poolRequest = CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build() ;

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

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

See the complete example on GitHub.

Add an app client

To enable the hosted web sign-up or sign-in UI for your app, create an app client. To create an app client, start by building a CreateUserPoolClientRequest object, with the name of the client as the value of its `clientName()`. Set `userPoolId()` to the ID of the user pool to which you want to attach this app client. Call the `createUserPoolClient()` method of your `CognitoIdentityProviderClient`, passing in the `CreateUserPoolClientRequest` object. You can capture the result of this request as a CreateUserPoolClientResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;
```

Code

```
    public static void createPoolClient ( CognitoIdentityProviderClient cognitoClient,
                                          String clientName,
                                          String userPoolId ) {
```

```
        try {

            CreateUserPoolClientResponse response = cognitoClient.createUserPoolClient(
                    CreateUserPoolClientRequest.builder()
                            .clientName(clientName)
                            .userPoolId(userPoolId)
                            .build()
            );

            System.out.println("User pool " + response.userPoolClient().clientName() + "
created. ID: " + response.userPoolClient().clientId());

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

See the complete example on GitHub.

Add a third-party identity provider

Adding an external identity provider (IdP) enables your users to log into your app using that service's login mechanism. To add a third-party IdP, start by building an UpdateIdentityPoolRequest object, with the name of the identity pool as the value of its `identityPoolName()`. Set `allowUnauthenticatedIdentities()` to `true` or `false`, specify the `identityPoolId()`, and define which login providers will be supported with `supportedLoginProviders()`. Call the `updateIdentityPool()` method of your `CognitoIdentityClient`, passing in the `UpdateIdentityPoolRequest` object. You can capture the result of this request as an UpdateIdentityPoolResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import software.amazon.awssdk.services.cognitoidentity.model.CognitoIdentityProvider;
import software.amazon.awssdk.services.cognitoidentity.model.UpdateIdentityPoolRequest;
import software.amazon.awssdk.services.cognitoidentity.model.UpdateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import java.util.ArrayList;
import java.util.List;
```

Code

```
    public static void createNewUser(CognitoIdentityProviderClient cognitoClient,
                                     String userPoolId,
                                     String name,
                                     String email,
                                     String password){

        try{

            AttributeType userAttrs = AttributeType.builder()
                    .name("email")
                    .value(email)
                    .build();

            AdminCreateUserRequest userRequest = AdminCreateUserRequest.builder()
                    .userPoolId(userPoolId)
                    .username(name)
```

```
                    .temporaryPassword(password)
                    .userAttributes(userAttrs)
                    .messageAction("SUPPRESS")
                    .build() ;

            AdminCreateUserResponse response = cognitoClient.adminCreateUser(userRequest);
            System.out.println("User " + response.user().username() + "is created. Status:
 " + response.user().userStatus());

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

See the complete example on GitHub.

Get credentials for an ID

To get the credentials for an identity in an identity pool, first build a GetCredentialsForIdentityRequest with the identity ID as the value of its `identityId()`. Call the `getCredentialsForIdentity()` method of your `CognitoIdentityClient`, passing in the `GetCredentialsForIdentityRequest`. You can capture the result of this request as a GetCredentialsForIdentityResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
```

Code

```
    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient, String
 identityId) {

        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest.builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response =
 cognitoClient.getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println("Identity ID " + response.identityId() + ", Access key ID "
 + response.credentials().accessKeyId());

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

See the complete example on GitHub.

For more information, see the Amazon Cognito Developer Guide.

Working with Amazon Kinesis Data Firehose

Amazon Kinesis Data Firehose provides a simple way to capture, transform, and load streaming data. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon Forecast

Amazon Forecast is a fully managed service for time-series forecasting. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Kinesis

This section provides examples of programming Amazon Kinesis using the AWS SDK for Java 2.x.

For more information about Kinesis, see the Amazon Kinesis Developer Guide.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Subscribing to Amazon Kinesis Data Streams (p. 192)

Subscribing to Amazon Kinesis Data Streams

The following examples show you how to retrieve and process data from Amazon Kinesis Data Streams using the `subscribeToShard` method. Kinesis Data Streams now employs the enhanced fanout feature and a low-latency HTTP/2 data retrieval API, making it easier for developers to run multiple low-latency, high-performance applications on the same Kinesis Data Stream.

Set up

First, create an asynchronous Kinesis client and a SubscribeToShardRequest object. These objects are used in each of the following examples to subscribe to Kinesis events.

Imports

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicInteger;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEventStream;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponse;
```

```
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
```

Code

```
        Region region = Region.US_EAST_1;
        KinesisAsyncClient client = KinesisAsyncClient.builder()
        .region(region)
        .build();

        SubscribeToShardRequest request = SubscribeToShardRequest.builder()
                .consumerARN(CONSUMER_ARN)
                .shardId("arn:aws:kinesis:us-east-1:111122223333:stream/StockTradeStream")
                .startingPosition(s -> s.type(ShardIteratorType.LATEST)).build();
```

Use the builder interface

You can use the `builder` method to simplify the creation of the SubscribeToShardResponseHandler.

Using the builder, you can set each lifecycle callback with a method call instead of implementing the full interface.

Code

```
   private static CompletableFuture<Void> responseHandlerBuilder(KinesisAsyncClient
client, SubscribeToShardRequest request) {
        SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
                .onComplete(() -> System.out.println("All records stream successfully"))
                // Must supply some type of subscriber
                .subscriber(e -> System.out.println("Received event - " + e))
                .build();
        return client.subscribeToShard(request, responseHandler);
   }
```

For more control of the publisher, you can use the `publisherTransformer` method to customize the publisher.

Code

```
   private static CompletableFuture<Void>
responseHandlerBuilderPublisherTransformer(KinesisAsyncClient client,
SubscribeToShardRequest request) {
        SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
                .publisherTransformer(p -> p.filter(e -> e instanceof
SubscribeToShardEvent).limit(100))
                .subscriber(e -> System.out.println("Received event - " + e))
                .build();
        return client.subscribeToShard(request, responseHandler);
   }
```

See the complete example on GitHub.

Use a custom response handler

For full control of the subscriber and publisher, implement the SubscribeToShardResponseHandler interface.

In this example, you implement the `onEventStream` method, which allows you full access to the publisher. This demonstrates how to transform the publisher to event records for printing by the subscriber.

Code

```
   private static CompletableFuture<Void> responseHandlerBuilderClassic(KinesisAsyncClient
client, SubscribeToShardRequest request) {
       SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {

           @Override
           public void responseReceived(SubscribeToShardResponse response) {
               System.out.println("Receieved initial response");
           }

           @Override
           public void onEventStream(SdkPublisher<SubscribeToShardEventStream> publisher)
{
               publisher
                       // Filter to only SubscribeToShardEvents
                       .filter(SubscribeToShardEvent.class)
                       // Flat map into a publisher of just records
                       .flatMapIterable(SubscribeToShardEvent::records)
                       // Limit to 1000 total records
                       .limit(1000)
                       // Batch records into lists of 25
                       .buffer(25)
                       // Print out each record batch
                       .subscribe(batch -> System.out.println("Record Batch - " + batch));
           }

           @Override
           public void complete() {
               System.out.println("All records stream successfully");
           }

           @Override
           public void exceptionOccurred(Throwable throwable) {
               System.err.println("Error during stream - " + throwable.getMessage());
           }
       };
       return client.subscribeToShard(request, responseHandler);
   }
```

See the complete example on GitHub.

Use the visitor interface

You can use a Visitor object to subscribe to specific events you're interested in watching.

Code

```
   private static CompletableFuture<Void>
responseHandlerBuilderVisitorBuilder(KinesisAsyncClient client, SubscribeToShardRequest
request) {
       SubscribeToShardResponseHandler.Visitor visitor =
SubscribeToShardResponseHandler.Visitor
               .builder()
               .onSubscribeToShardEvent(e -> System.out.println("Received subscribe to
shard event " + e))
               .build();
       SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
```

```
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
                .subscriber(visitor)
                .build();
        return client.subscribeToShard(request, responseHandler);
    }
```

See the complete example on GitHub.

Use a custom subscriber

You can also implement your own custom subscriber to subscribe to the stream.

This code snippet shows an example subscriber.

Code

```
    private static class MySubscriber implements Subscriber<SubscribeToShardEventStream> {

        private Subscription subscription;
        private AtomicInteger eventCount = new AtomicInteger(0);

        @Override
        public void onSubscribe(Subscription subscription) {
            this.subscription = subscription;
            this.subscription.request(1);
        }

        @Override
        public void onNext(SubscribeToShardEventStream shardSubscriptionEventStream) {
            System.out.println("Received event " + shardSubscriptionEventStream);
            if (eventCount.incrementAndGet() >= 100) {
                // You can cancel the subscription at any time if you wish to stop
receiving events.
                subscription.cancel();
            }
            subscription.request(1);
        }

        @Override
        public void onError(Throwable throwable) {
            System.err.println("Error occurred while stream - " + throwable.getMessage());
        }

        @Override
        public void onComplete() {
            System.out.println("Finished streaming all events");
        }
    }
```

You can pass that custom subscriber to the `subscribe` method, similarly to preview examples. The following code snippet shows this example.

Code

```
    private static CompletableFuture<Void>
responseHandlerBuilderSubscriber(KinesisAsyncClient client, SubscribeToShardRequest
request) {
        SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
                .builder()
                .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
```

```
                .subscriber(MySubscriber::new)
                .build();
        return client.subscribeToShard(request, responseHandler);
    }
```

See the complete example on GitHub.

Write data records into a Kinesis data stream

You can use the AmazonKinesisClient object to write data records into a Kinesis data stream by using the `putRecords` method. To successfully invoke this method, create a PutRecordsRequest object. You pass the name of the data steam to the `streamName` method. Also you must pass the data by using the `putRecords` method (as shown in the following code example).

Imports

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
```

In the following Java code example, notice that **StockTrade** object is used as the data to write to the Kinesis data stream. Before running this example, ensure that you have created the data stream.

Code

```
    public static void setStockData( KinesisClient kinesisClient, String streamName) {

        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x=0; x<index; x++){
                StockTrade trade = stockTradeGenerator.getRandomTrade();
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }

    private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
                                       String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization by the
 Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }
```

```
        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
            .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as the
 partition key, explained in the Supplemental Information section below.
            .streamName(streamName)
            .data(SdkBytes.fromByteArray(bytes))
            .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String streamName) {
        try {
            DescribeStreamRequest describeStreamRequest = DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if(!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE")) {
                System.err.println("Stream " + streamName + " is not active. Please wait a
 few moments and try again.");
                System.exit(1);
            }

        }catch (KinesisException e) {
            System.err.println("Error found while describing the stream " + streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Use a third-party library

You can use other third-party libraries instead of implementing a custom subscriber. This example demonstrates using the RxJava implementation, but you can use any library that implements the Reactive Streams interfaces. See the RxJava wiki page on Github for more information on that library.

To use the library, add it as a dependency. If you're using Maven, the example shows the POM snippet to use.

POM Entry

```
<dependency>
 <groupId>io.reactivex.rxjava2</groupId>
 <artifactId>rxjava</artifactId>
 <version>2.1.14</version>
</dependency>
```

Imports

```
import java.net.URI;
import java.util.concurrent.CompletableFuture;
```

```
import io.reactivex.Flowable;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.SdkHttpConfigurationOption;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.awssdk.services.kinesis.model.ShardIteratorType;
import software.amazon.awssdk.services.kinesis.model.StartingPosition;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardEvent;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardRequest;
import software.amazon.awssdk.services.kinesis.model.SubscribeToShardResponseHandler;
import software.amazon.awssdk.utils.AttributeMap;
```

This example uses RxJava in the `onEventStream` lifecycle method. This gives you full access to the publisher, which can be used to create an Rx Flowable.

Code

```
        SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
            .builder()
            .onError(t -> System.err.println("Error during stream - " + t.getMessage()))
            .onEventStream(p -> Flowable.fromPublisher(p)
                                        .ofType(SubscribeToShardEvent.class)
                                        .flatMapIterable(SubscribeToShardEvent::records)
                                        .limit(1000)
                                        .buffer(25)
                                        .subscribe(e -> System.out.println("Record batch =
 " + e)))
            .build();
```

You can also use the `publisherTransformer` method with the `Flowable` publisher. You must adapt the `Flowable` publisher to an *SdkPublisher*, as shown in the following example.

Code

```
        SubscribeToShardResponseHandler responseHandler = SubscribeToShardResponseHandler
            .builder()
            .onError(t -> System.err.println("Error during stream - " + t.getMessage()))
            .publisherTransformer(p ->
 SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
            .build();
```

See the complete example on GitHub.

More information

- SubscribeToShardEvent in the Amazon Kinesis API Reference
- SubscribeToShard in the Amazon Kinesis API Reference

Invoke, list, and delete AWS Lambda functions

This section provides examples of programming with the Lambda service client by using the AWS SDK for Java 2.x.

Topics

- Invoke a Lambda function (p. 199)

- List Lambda functions (p. 199)
- Delete a Lambda function (p. 200)

Invoke a Lambda function

You can invoke a Lambda function by creating a LambdaClient object and invoking its `invoke` method. Create an InvokeRequest object to specify additional information such as the function name and the payload to pass to the Lambda function. Function names appear as *arn:aws:lambda:us-east-1:123456789012:function:HelloFunction*. You can retrieve the value by looking at the function in the AWS Management Console.

To pass payload data to a function, create a SdkBytes object that contains information. For example, in the following code example, notice the JSON data passed to the Lambda function.

Imports

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

The following code example demonstrates how to invoke a Lambda function.

```
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {

         InvokeResponse res = null ;
        try {
            //Need a SdkBytes instance for the payload
            String json = "{\"Hello \":\"Paris\"}";
            SdkBytes payload = SdkBytes.fromUtf8String(json) ;

            //Setup an InvokeRequest
            InvokeRequest request = InvokeRequest.builder()
                    .functionName(functionName)
                    .payload(payload)
                    .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String() ;
            System.out.println(value);

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

List Lambda functions

Build a LambdaClient object and invoke its `listFunctions` method. This method returns a ListFunctionsResponse object. You can invoke this object's `functions` method to return a list of FunctionConfiguration objects. You can iterate through the list to retrieve information about the functions. For example, the following Java code example shows how to get each function name.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
import java.util.List;
```

Code

The following Java code example demonstrates how to retrieve a list of function names.

```
    public static void listFunctions(LambdaClient awsLambda) {

        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();

            for (FunctionConfiguration config: list) {
                System.out.println("The function name is "+config.functionName());
            }

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Delete a Lambda function

Build a LambdaClient object and invoke its `deleteFunction` method. Create a DeleteFunctionRequest object and pass it to the `deleteFunction` method. This object contains information such as the name of the function to delete. Function names appear as *arn:aws:lambda:us-east-1:123456789012:function:HelloFunction*. You can retrieve the value by looking at the function in the AWS Management Console.

Imports

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;
```

Code

The following Java code demonstrates how to delete a Lambda function.

```
    public static void deleteLambdaFunction(LambdaClient awsLambda, String functionName ) {
        try {
            //Setup an DeleteFunctionRequest
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("The "+functionName +" function was deleted");
```

```
        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Working with AWS Elemental MediaConvert

AWS Elemental MediaConvert is a file-based video processing service that allows video providers to transcode content for broadcast and multiscreen delivery. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with AWS Elemental MediaStore

AWS Elemental MediaStore is an AWS storage service optimized for media. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with AWS Migration Hub

AWS Migration Hub provides a single place to monitor migrations in any AWS region where your migration tools are available. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon Pinpoint

You can use Amazon Pinpoint to send relevant, personalized messages to your customers via multiple communication channels, such as push notifications, SMS, and email.

Create a project

A project (or application) in Amazon Pinpoint is a collection of settings, customer data, segments, and campaigns.

To create a project, start by building a CreateApplicationRequest object with the name of the project as the value of its `name()`. Then build a CreateAppRequest object, passing in the `CreateApplicationRequest` object as the value of its `createApplicationRequest()` method. Call the `createApp()` method of your PinpointClient, passing in the `CreateAppRequest` object. Capture the result of this request as a CreateAppResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

Code

```
    public static String createApplication(PinpointClient pinpoint, String appName) {

        try {
            CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
                    .name(appName)
                    .build();

            CreateAppRequest request = CreateAppRequest.builder()
                    .createApplicationRequest(appRequest)
                    .build();

            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

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

See the complete example on GitHub.

Create a dynamic segment

A segment is a set of customers who share specific attributes, such as the city they live in or how frequently they visit your website. A dynamic segment is one that's based on attributes that you define, and can change over time.

To create a dynamic segment, first build all of the dimensions you want for this segment. For example, the following code snippet is set to include customers who were active on the site in the last 30 days. You can do this by first building a RecencyDimension object with the `duration()` and `recencyType()` you want (that is, `ACTIVE` or `INACTIVE`), and then passing this object to a SegmentBehaviors builder object as the value of `recency()`.

When you have defined your segment attributes, build them into a SegmentDimensions object. Then build a WriteSegmentRequest object, passing in the `SegmentDimensions` object as the value of its `dimensions()`. Next, build a CreateSegmentRequest object, passing in the `WriteSegmentRequest` object as the value of its `writeSegmentRequest()`. Finally, call the `createSegment()` method of your `PinpointClient`, passing in the `CreateSegmentRequest` object. Capture the result of this request as a CreateSegmentResponse object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;
```

Code

```
    public static SegmentResponse createSegment(PinpointClient client, String appId) {

        try {
            Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
            segmentAttributes.put("Team", AttributeDimension.builder()
                    .attributeType(AttributeType.INCLUSIVE)
                    .values("Lakers")
                    .build());

            RecencyDimension recencyDimension = RecencyDimension.builder()
                    .duration("DAY_30")
                    .recencyType("ACTIVE")
                    .build();

            SegmentBehaviors segmentBehaviors = SegmentBehaviors.builder()
                    .recency(recencyDimension)
                    .build();

            SegmentDemographics segmentDemographics = SegmentDemographics
                    .builder()
                    .build();

            SegmentLocation segmentLocation = SegmentLocation
                    .builder()
                    .build();

            SegmentDimensions dimensions = SegmentDimensions
                    .builder()
                    .attributes(segmentAttributes)
                    .behavior(segmentBehaviors)
                    .demographic(segmentDemographics)
                    .location(segmentLocation)
                    .build();

            WriteSegmentRequest writeSegmentRequest = WriteSegmentRequest.builder()
                    .name("MySegment")
                    .dimensions(dimensions)
                    .build();

            CreateSegmentRequest createSegmentRequest = CreateSegmentRequest.builder()
                    .applicationId(appId)
                    .writeSegmentRequest(writeSegmentRequest)
                    .build();

            CreateSegmentResponse createSegmentResult =
 client.createSegment(createSegmentRequest);
            System.out.println("Segment ID: " +
 createSegmentResult.segmentResponse().id());
            System.out.println("Done");
            return createSegmentResult.segmentResponse();

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

See the complete example on GitHub.

Import a static segment

A static segment is one you create and import from outside of Amazon Pinpoint. The following example code shows how to create a static segment by importing it from Amazon S3.

Prerequisite

Before you can complete this example, you need to create an IAM role that grants Amazon Pinpoint access to Amazon S3. For more information, see IAM role for importing endpoints or segments in the Amazon Pinpoint Developer Guide.

To import a static segment, start by building an ImportJobRequest object. In the builder, specify the `s3Url()`, `roleArn()`, and `format()`.

> **Note**
> For more information about the properties of an `ImportJobRequest`, see the ImportJobRequest section of Import Jobs in the Amazon Pinpoint API Reference.

Then build a CreateImportJobRequest object, passing in the `ImportJobRequest` object as the value of its `importJobRequest()`, and the `ID` of your project as the `applicationId()`. Call the `createImportJob()` method of your `PinpointClient`, passing in the `CreateImportJobRequest` object. Capture the result of this request as a `CreateImportJobResponse` object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

Code

```
    public static ImportJobResponse createImportSegment(PinpointClient client,
                                                        String appId,
                                                        String bucket,
                                                        String key,
                                                        String roleArn) {

        try {
             ImportJobRequest importRequest = ImportJobRequest.builder()
                     .defineSegment(true)
                     .registerEndpoints(true)
                     .roleArn(roleArn)
                     .format(Format.JSON)
                     .s3Url("s3://" + bucket + "/" + key)
                     .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                     .importJobRequest(importRequest)
                     .applicationId(appId)
                     .build();

            CreateImportJobResponse jobResponse = client.createImportJob(jobRequest);
```

```
            return jobResponse.importJobResponse();

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

See the complete example on GitHub.

List segments for your project

To list the segments associated with a particular project, start by building a GetSegmentsRequest object, with the `ID` of the project as the value of its `applicationId()`. Next, call the `getSegments()` method of your `PinpointClient`, passing in the `GetSegmentsRequest` object. Capture the result of this request as a GetSegmentsResponse object. Finally, instantiate a List object upcasted to the SegmentResponse class. Then call the `segmentsResponse().item()` of `GetSegmentsResponse`, as demonstrated in the following code snippet. From there, you can iterate through the results.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;
```

Code

```
    public static void listSegs( PinpointClient pinpoint, String appId) {

        try {
            GetSegmentsRequest request = GetSegmentsRequest.builder()
                    .applicationId(appId)
                    .build();

            GetSegmentsResponse response = pinpoint.getSegments(request);
            List<SegmentResponse> segments = response.segmentsResponse().item();

            for(SegmentResponse segment: segments) {
                System.out.println("Segement " + segment.id() + " " + segment.name() + " "
 + segment.lastModifiedDate());
            }
        } catch ( PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

See the complete example on GitHub.

Create a campaign

A campaign is an initiative meant to engage a particular audience segment by sending messages to those customers.

To create a campaign, first build all of the settings you want for this campaign. In the following code snippet, for example, the campaign will start immediately because the `startTime()` of the Schedule is set to `IMMEDIATE`. To set it to start at a specific time instead, specify a time in ISO 8601 format.

> **Note**
> For more information about the settings available for campaigns, see the **Schedule** section of Campaigns in the Amazon Pinpoint API Reference.

After you define your campaign configuration, build it into a WriteCampaignRequest object. None of the methods of the `builder()` of the `WriteCampaignRequest` are required. But you do need to include any of the configuration settings (MessageConfiguration) that you set for the campaign. We also recommend that you include a `name` and a `description` for your campaign so you can easily distinguish it from other campaigns. Call the `createCampaign ()` method of your `PinpointClient`, passing in the `WriteCampaignRequest` object. Capture the result of this request as a CreateCampaignResponse object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

Code

```
    public static void createPinCampaign(PinpointClient pinpoint, String appId, String
 segmentId) {

        CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
        System.out.println("Campaign " + result.name() + " created.");
        System.out.println(result.description());

    }

    public static CampaignResponse createCampaign(PinpointClient client, String appID,
 String segmentID) {

        try {
            Schedule schedule = Schedule.builder()
                    .startTime("IMMEDIATE")
                    .build();

            Message defaultMessage = Message.builder()
                    .action(Action.OPEN_APP)
                    .body("My message body.")
                    .title("My message title.")
                    .build();

            MessageConfiguration messageConfiguration = MessageConfiguration.builder()
                    .defaultMessage(defaultMessage)
                    .build();

            WriteCampaignRequest request = WriteCampaignRequest.builder()
                    .description("My description")
                    .schedule(schedule)
```

```
                    .name("MyCampaign")
                    .segmentId(segmentID)
                    .messageConfiguration(messageConfiguration)
                    .build();

            CreateCampaignResponse result = client.createCampaign(
                    CreateCampaignRequest.builder()
                            .applicationId(appID)
                            .writeCampaignRequest(request).build()
            );

            System.out.println("Campaign ID: " + result.campaignResponse().id());

            return result.campaignResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

See the complete example on GitHub.

Send a message

To send an SMS text message through Amazon Pinpoint, first build an AddressConfiguration object to specify the `channelType()`. (In the following example, it's set to `ChannelType.SMS` to indicate the message will be sent via SMS.) Initialize a HashMap to store the destination phone number and the `AddressConfiguration` object. Next, build an SMSMessage object containing the relevant values. These include the `originationNumber`, the type of message (`messageType`), and the body of the message itself.

When you have created the message, build the `SMSMessage` object into a DirectMessageConfiguration object. Build your Map object and `DirectMessageConfiguration` object into a MessageRequest object. Build a SendMessagesRequest object, including your project ID (`applicationId`) and your `MessageRequest` object. Call the `sendMessages()` method of your `PinpointClient`, passing in the `SendMessagesRequest` object. Capture the result of this request as a SendMessagesResponse object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;
```

Code

```
    public static void sendSMSMessage(PinpointClient pinpoint, String message, String
 appId, String originationNumber, String destinationNumber) {
```

```
    try {

        Map<String, AddressConfiguration> addressMap =
                new HashMap<String, AddressConfiguration>();

        AddressConfiguration addConfig = AddressConfiguration.builder()
                .channelType(ChannelType.SMS)
                .build();

        addressMap.put(destinationNumber, addConfig);

        SMSMessage smsMessage = SMSMessage.builder()
                .body(message)
                .messageType(messageType)
                .originationNumber(originationNumber)
                .senderId(senderId)
                .keyword(registeredKeyword)
                .build();

        // Create a DirectMessageConfiguration object
        DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
                .smsMessage(smsMessage)
                .build();

        MessageRequest msgReq = MessageRequest.builder()
                .addresses(addressMap)
                .messageConfiguration(direct)
                .build();

        // create a  SendMessagesRequest object
        SendMessagesRequest request = SendMessagesRequest.builder()
                .applicationId(appId)
                .messageRequest(msgReq)
                .build();

        SendMessagesResponse response= pinpoint.sendMessages(request);

        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        //Write out the result of sendMessage
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

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

See the complete example on GitHub.

For more information, see the Amazon Pinpoint Developer Guide.

Working with Amazon Polly

Amazon Polly is a service that turns text into lifelike speech, allowing you to create applications that talk, and build entirely new categories of speech-enabled functionality. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon SageMaker

Amazon SageMaker is a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon Simple Notification Service

With Amazon Simple Notification Service, you can easily push real-time notification messages from your applications to subscribers over multiple communication channels. This topic describes how to perform some of the basic functions of Amazon SNS.

Create a topic

A **topic** is a logical grouping of communication channels that defines which systems to send a message to, for example, fanning out a message to AWS Lambda and an HTTP webhook. You send messages to Amazon SNS, then they're distributed to the channels defined in the topic. This makes the messages available to subscribers.

To create a topic, first build a CreateTopicRequest object, with the name of the topic set using the `name()` method in the builder. Then, send the request object to Amazon SNS by using the `createTopic()` method of the SnsClient. You can capture the result of this request as a CreateTopicResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static String createSNSTopic(SnsClient snsClient, String topicName ) {

        CreateTopicResponse result = null;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();
        } catch (SnsException e) {

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

See the complete example on GitHub.

List your Amazon SNS topics

To retrieve a list of your existing Amazon SNS topics, build a ListTopicsRequest object. Then, send the request object to Amazon SNS by using the `listTopics()` method of the `SnsClient`. You can capture the result of this request as a ListTopicsResponse object.

The following code snippet prints out the HTTP status code of the request and a list of Amazon Resource Names (ARNs) for your Amazon SNS topics.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static void listSNSTopics(SnsClient snsClient) {

        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                   .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

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

See the complete example on GitHub.

Subscribe an endpoint to a topic

After you create a topic, you can configure which communication channels will be endpoints for that topic. Messages are distributed to these endpoints after Amazon SNS receives them.

To configure a communication channel as an endpoint for a topic, subscribe that endpoint to the topic. To start, build a SubscribeRequest object. Specify the communication channel (for example, `lambda` or `email`) as the `protocol()`. Set the `endpoint()` to the relevant output location (for example, the ARN of a Lambda function or an email address), and then set the ARN of the topic to which you want to subscribe as the `topicArn()`. Send the request object to Amazon SNS by using the `subscribe()` method of the `SnsClient`. You can capture the result of this request as a SubscribeResponse object.

The following code snippet shows how to subscribe an email address to a topic.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

Code

```
    public static void subEmail(SnsClient snsClient, String topicArn, String email) {

        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("email")
                .endpoint(email)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n
 Status is " + result.sdkHttpResponse().statusCode());

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

See the complete example on GitHub.

Publish a message to a topic

After you have a topic and one or more endpoints configured for it, you can publish a message to it. To start, build a PublishRequest object. Specify the `message()` to send, and the ARN of the topic (`topicArn()`) to send it to. Then, send the request object to Amazon SNS by using the `publish()` method of the `SnsClient`. You can capture the result of this request as a PublishResponse object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out.println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

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

See the complete example on GitHub.

Unsubscribe an endpoint from a topic

You can remove the communication channels configured as endpoints for a topic. After doing that, the topic itself continues to exist and distribute messages to any other endpoints configured for that topic.

To remove a communication channel as an endpoint for a topic, unsubscribe that endpoint from the topic. To start, build an UnsubscribeRequest object and set the ARN of the topic you want to unsubscribe from as the `subscriptionArn()`. Then send the request object to SNS by using the `unsubscribe()` method of the `SnsClient`. You can capture the result of this request as an UnsubscribeResponse object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

Code

```
    public static void unSub(SnsClient snsClient, String subscriptionArn) {

        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);

            System.out.println("\n\nStatus was " + result.sdkHttpResponse().statusCode()
                + "\n\nSubscription was removed for " + request.subscriptionArn());

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

See the complete example on GitHub.

Delete a topic

To delete an Amazon SNS topic, first build a DeleteTopicRequest object with the ARN of the topic set as the `topicArn()` method in the builder. Then send the request object to Amazon SNS by using the `deleteTopic()` method of the `SnsClient`. You can capture the result of this request as a DeleteTopicResponse object, as demonstrated in the following code snippet.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

Code

```
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

        try {
```

```
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " + result.sdkHttpResponse().statusCode());

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

See the complete example on GitHub.

For more information, see the Amazon Simple Notification Service Developer Guide.

Working with Amazon Simple Queue Service

This section provides examples of programming Amazon Simple Queue Service using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Working with Amazon Simple Queue Service message queues (p. 213)
- Sending, receiving, and deleting Amazon Simple Queue Service messages (p. 215)

Working with Amazon Simple Queue Service message queues

A *message queue* is the logical container used for sending messages reliably in Amazon Simple Queue Service. There are two types of queues: *standard* and *first-in, first-out* (FIFO). To learn more about queues and the differences between these types, see the Amazon Simple Queue Service Developer Guide.

This topic describes how to create, list, delete, and get the URL of an Amazon Simple Queue Service queue by using the AWS SDK for Java.

Create a queue

Use the SqsClient's `createQueue` method, and provide a CreateQueueRequest object that describes the queue parameters.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);
```

See the complete sample on GitHub.

List queues

To list the Amazon Simple Queue Service queues for your account, call the SqsClient's `listQueues` method with a ListQueuesRequest object.

Using the `listQueues` overload without any parameters returns *all queues*, up to 1,000 queues . You can supply a queue name prefix to the `ListQueuesRequest` object to limit the results to queues that match that prefix.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
 ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
 sqsClient.listQueues(listQueuesRequest);

            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

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

See the complete sample on GitHub.

Get the URL for a queue

Call the SqsClient's `getQueueUrl` method. with a GetQueueUrlRequest object.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            GetQueueUrlResponse getQueueUrlResponse =

 sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            String queueUrl = getQueueUrlResponse.queueUrl();
            return queueUrl;

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

```
        }
        return "";
```

See the complete sample on GitHub.

Delete a queue

Provide the queue's URL (p. 214) to the DeleteMessageRequest object. Then call the SqsClient's `deleteQueue` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

        try {

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

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

See the complete sample on GitHub.

More information

- How Amazon Simple Queue Service Queues Work in the Amazon Simple Queue Service Developer Guide
- CreateQueue in the Amazon Simple Queue Service API Reference
- GetQueueUrl in the Amazon Simple Queue Service API Reference
- ListQueues in the Amazon Simple Queue Service API Reference
- DeleteQueues in the Amazon Simple Queue Service API Reference

Sending, receiving, and deleting Amazon Simple Queue Service messages

A message is a piece of data that can be sent and received by distributed components. Messages are always delivered using an SQS Queue (p. 213).

Send a message

Add a single message to an Amazon Simple Queue Service queue by calling the SqsClient client `sendMessage` method. Provide a SendMessageRequest object that contains the queue's URL (p. 214), message body, and optional delay value (in seconds).

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            sqsClient.sendMessage(SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello world!")
                .delaySeconds(10)
                .build());
```

Send multiple messages in a request

Send more than one message in a single request by using the SqsClient `sendMessageBatch` method. This method takes a SendMessageBatchRequest that contains the queue URL and a list of messages to send. (Each message is a SendMessageBatchRequestEntry.) You can also delay sending a specific message by setting a delay value on the message.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            SendMessageBatchRequest sendMessageBatchRequest =
 SendMessageBatchRequest.builder()
                .queueUrl(queueUrl)

 .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
 1").build(),
                         SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
 2").delaySeconds(10).build())
                .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

See the complete sample on GitHub.

Retrieve Messages

Retrieve any messages that are currently in the queue by calling the SqsClient `receiveMessage` method. This method takes a ReceiveMessageRequest that contains the queue URL. You can also specify the maximum number of messages to return. Messages are returned as a list of Message objects.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
            ReceiveMessageRequest receiveMessageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .maxNumberOfMessages(5)
                .build();
            List<Message> messages =
 sqsClient.receiveMessage(receiveMessageRequest).messages();
            return messages;
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

Delete a message after receipt

After receiving a message and processing its contents, delete the message from the queue by sending the message's receipt handle and queue URL to the SqsClient `deleteMessage` method.

Imports

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Code

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest = DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .receiptHandle(message.receiptHandle())
                    .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
```

See the complete sample on GitHub.

More Info

- How Amazon Simple Queue Service Queues Work in the Amazon Simple Queue Service Developer Guide
- SendMessage in the Amazon Simple Queue Service API Reference
- SendMessageBatch in the Amazon Simple Queue Service API Reference
- ReceiveMessage in the Amazon Simple Queue Service API Reference
- DeleteMessage in the Amazon Simple Queue Service API Reference

Working with AWS Systems Manager

AWS Systems Manager is Amazon software that can be installed and configured on an Amazon EC2 instance, an on-premises server, or a virtual machine (VM). See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon Simple Workflow Service

The Amazon Simple Workflow Service makes it easy to build applications that coordinate work across distributed components. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon Transcribe

This section provides examples of programming Amazon Transcribe using the AWS SDK for Java 2.x.

The following examples include only the code needed to demonstrate each technique. The complete example code is available on GitHub. From there, you can download a single source file or clone the repository locally to get all the examples to build and run.

Topics

- Working with Amazon Transcribe (p. 218)

Working with Amazon Transcribe

The following example shows how bidirectional streaming works using Amazon Transcribe. Bidirectional streaming implies that there's both a stream of data going to the service and being received back in real time. The example uses Amazon Transcribe streaming transcription to send an audio stream and receive a stream of transcribed text back in real time.

See Streaming Transcription in the Amazon Transcribe Developer Guide to learn more about this feature.

See Getting Started in the Amazon Transcribe Developer Guide to get started using Amazon Transcribe.

Set up the microphone

This code uses the javax.sound.sampled package to stream audio from an input device.

Code

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;

public class Microphone {

    public static TargetDataLine get() throws Exception {
        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
        DataLine.Info datalineInfo = new DataLine.Info(TargetDataLine.class, format);

        TargetDataLine dataLine = (TargetDataLine) AudioSystem.getLine(datalineInfo);
        dataLine.open(format);

        return dataLine;
```

```
    }
}
```

See the complete example on GitHub.

Create a publisher

This code implements a publisher that publishes audio data from the Amazon Transcribe audio stream.

Code

```
package com.amazonaws.transcribe;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicLong;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.transcribestreaming.model.AudioEvent;
import software.amazon.awssdk.services.transcribestreaming.model.AudioStream;
import
 software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException;


public class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;

    public AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        s.onSubscribe(new SubscriptionImpl(s, inputStream));
    }

    private class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;

        private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
            }

            demand.getAndAdd(n);
```

```
            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (TranscribeStreamingException e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {

        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

See the complete example on GitHub.

Create the client and start the stream

In the main method, create a request object, start the audio input stream and instantiate the publisher with the audio input.

You must also create a StartStreamTranscriptionResponseHandler to specify how to handle the response from Amazon Transcribe.

Then, use the TranscribeStreamingAsyncClient's `startStreamTranscription` method to start the bidirectional streaming.

Imports

```
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
 software.amazon.awssdk.services.transcribestreaming.model.TranscribeStreamingException ;
import
 software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionRequest;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import
 software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponseHandler;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

Code

```
    public static void convertAudio(TranscribeStreamingAsyncClient client) throws Exception
 {

        try {

            StartStreamTranscriptionRequest request =
 StartStreamTranscriptionRequest.builder()
                    .mediaEncoding(MediaEncoding.PCM)
                    .languageCode(LanguageCode.EN_US)
                    .mediaSampleRateHertz(16_000).build();

            TargetDataLine mic = Microphone.get();
            mic.start();

            AudioStreamPublisher publisher = new AudioStreamPublisher(new
 AudioInputStream(mic));

            StartStreamTranscriptionResponseHandler response =
                    StartStreamTranscriptionResponseHandler.builder().subscriber(e -> {
                        TranscriptEvent event = (TranscriptEvent) e;
                        event.transcript().results().forEach(r ->
 r.alternatives().forEach(a -> System.out.println(a.transcript())));
                    }).build();

            // Keeps Streaming until you end the Java program
            client.startStreamTranscription(request, publisher, response);

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

See the complete example on GitHub.

More information

- How It Works in the Amazon Transcribe Developer Guide.
- Getting Started With Streaming Audio in the Amazon Transcribe Developer Guide.
- Guidelines and Limits in the Amazon Transcribe Developer Guide.

Working with Amazon Translate

Amazon Translate removes the complexity of building real-time and batch translation capabilities into your applications. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

Working with Amazon WorkDocs

Amazon WorkDocs is a fully managed, secure content creation, storage, and collaboration service. See the following resources for complete code examples with instructions.

Link to Github

Link to AWS Code Sample Catalog

SDK for Java 2.x code examples

The code examples in this topic show you how to use the AWS SDK for Java 2.x with AWS.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Cross-service examples are sample applications that work across multiple AWS services.

Examples

- Actions and scenarios using SDK for Java 2.x (p. 222)
- Cross-service examples using SDK for Java 2.x (p. 656)

Actions and scenarios using SDK for Java 2.x

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS services.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Services

- API Gateway examples using SDK for Java 2.x (p. 223)
- Application Recovery Controller examples using SDK for Java 2.x (p. 226)
- Aurora examples using SDK for Java 2.x (p. 228)
- CloudFront examples using SDK for Java 2.x (p. 249)
- CloudWatch examples using SDK for Java 2.x (p. 259)
- CloudWatch Events examples using SDK for Java 2.x (p. 288)

- CloudWatch Logs examples using SDK for Java 2.x (p. 290)
- Amazon Cognito Identity Provider examples using SDK for Java 2.x (p. 293)
- Amazon Comprehend examples using SDK for Java 2.x (p. 304)
- DynamoDB examples using SDK for Java 2.x (p. 308)
- Amazon EC2 examples using SDK for Java 2.x (p. 337)
- Amazon EC2 Auto Scaling examples using SDK for Java 2.x (p. 359)
- Amazon ECS examples using SDK for Java 2.x (p. 373)
- EventBridge examples using SDK for Java 2.x (p. 377)
- AWS Glue examples using SDK for Java 2.x (p. 379)
- IAM examples using SDK for Java 2.x (p. 391)
- AWS KMS examples using SDK for Java 2.x (p. 408)
- Amazon Keyspaces examples using SDK for Java 2.x (p. 414)
- Kinesis examples using SDK for Java 2.x (p. 431)
- Lambda examples using SDK for Java 2.x (p. 435)
- MediaConvert examples using SDK for Java 2.x (p. 442)
- Amazon Personalize examples using SDK for Java 2.x (p. 452)
- Amazon Personalize Events examples using SDK for Java 2.x (p. 471)
- Amazon Personalize Runtime examples using SDK for Java 2.x (p. 473)
- Amazon Pinpoint examples using SDK for Java 2.x (p. 476)
- Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x (p. 490)
- Amazon RDS examples using SDK for Java 2.x (p. 491)
- Amazon Redshift examples using SDK for Java 2.x (p. 511)
- Amazon Rekognition examples using SDK for Java 2.x (p. 513)
- Route 53 domain registration examples using SDK for Java 2.x (p. 540)
- Amazon S3 examples using SDK for Java 2.x (p. 554)
- S3 Glacier examples using SDK for Java 2.x (p. 584)
- Amazon SES examples using SDK for Java 2.x (p. 591)
- Amazon SES API v2 examples using SDK for Java 2.x (p. 596)
- Amazon SNS examples using SDK for Java 2.x (p. 597)
- Amazon SQS examples using SDK for Java 2.x (p. 613)
- AWS STS examples using SDK for Java 2.x (p. 619)
- Secrets Manager examples using SDK for Java 2.x (p. 620)
- Step Functions examples using SDK for Java 2.x (p. 624)
- AWS Support examples using SDK for Java 2.x (p. 638)
- Amazon Textract examples using SDK for Java 2.x (p. 652)

API Gateway examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with API Gateway.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a REST API

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

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

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

- For API details, see CreateRestApi in *AWS SDK for Java 2.x API Reference*.

Delete a REST API

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

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

       try {
           DeleteRestApiRequest request = DeleteRestApiRequest.builder()
               .restApiId(restApiId)
```

```
                .build();

            apiGateway.deleteRestApi(request);
            System.out.println("The API was successfully deleted");

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

- For API details, see DeleteRestApi in *AWS SDK for Java 2.x API Reference*.

Delete a deployment

The following code example shows how to delete a deployment.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

            apiGateway.deleteDeployment(request);
            System.out.println("Deployment was deleted" );

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

- For API details, see DeleteDeployment in *AWS SDK for Java 2.x API Reference*.

Deploy a REST API

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

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

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

- For API details, see CreateDeployment in *AWS SDK for Java 2.x API Reference*.

Application Recovery Controller examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Application Recovery Controller.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Get the state of a routing control

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

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static GetRoutingControlStateResponse
 getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
                                                                        String
 routingControlArn) {
        // As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
        // For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
```

```
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region())).build();
                return client.getRoutingControlState(
                        GetRoutingControlStateRequest.builder()
                                .routingControlArn(routingControlArn).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- For API details, see GetRoutingControlState in *AWS SDK for Java 2.x API Reference*.

Update the state of a routing control

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

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static UpdateRoutingControlStateResponse
 updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
                                                                          String
 routingControlArn,
                                                                          String
 routingControlState) {
        // As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
        // For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region()))
                        .build();
                return client.updateRoutingControlState(
                        UpdateRoutingControlStateRequest.builder()

 .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- For API details, see UpdateRoutingControlState in *AWS SDK for Java 2.x API Reference*.

Aurora examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Aurora.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a DB cluster

The following code example shows how to create an Aurora DB cluster.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see CreateDBCluster in *AWS SDK for Java 2.x API Reference*.

Create a DB cluster parameter group

The following code example shows how to create an Aurora DB cluster parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see CreateDBClusterParameterGroup in *AWS SDK for Java 2.x API Reference*.

Create a DB cluster snapshot

The following code example shows how to create an Aurora DB cluster snapshot.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

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

- For API details, see CreateDBClusterSnapshot in *AWS SDK for Java 2.x API Reference*.

Create a DB instance in a DB cluster

The following code example shows how to create a DB instance in an Aurora DB cluster.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

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

- For API details, see CreateDBInstance in *AWS SDK for Java 2.x API Reference*.

Delete a DB cluster

The following code example shows how to delete an Aurora DB cluster.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

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

- For API details, see DeleteDBCluster in *AWS SDK for Java 2.x API Reference*.

Delete a DB cluster parameter group

The following code example shows how to delete an Aurora DB cluster parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest.builder()
```

```
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();

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

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

- For API details, see DeleteDBClusterParameterGroup in *AWS SDK for Java 2.x API Reference*.

Delete a DB instance

The following code example shows how to delete an Aurora DB instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see DeleteDBInstance in *AWS SDK for Java 2.x API Reference*.

Describe DB cluster parameter groups

The following code example shows how to describe Aurora DB cluster parameter groups.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see DescribeDBClusterParameterGroups in *AWS SDK for Java 2.x API Reference*.

Describe DB cluster snapshots

The following code example shows how to describe Aurora DB cluster snapshots.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

```
                }
            }
        }

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

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

- For API details, see DescribeDBClusterSnapshots in *AWS SDK for Java 2.x API Reference*.

Describe DB clusters

The following code example shows how to describe Aurora DB clusters.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see DescribeDBClusters in *AWS SDK for Java 2.x API Reference*.

Describe DB instances

The following code example shows how to describe Aurora DB instances.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

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

- For API details, see DescribeDBInstances in *AWS SDK for Java 2.x API Reference*.

Describe database engine versions

The following code example shows how to describe Aurora database engine versions.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

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

- For API details, see DescribeDBEngineVersions in *AWS SDK for Java 2.x API Reference*.

Describe options for DB instances

The following code example shows how to describe options for Aurora DB instances.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

           // Get all DBEngineVersion objects.
```

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

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

- For API details, see DescribeOrderableDBInstanceOptions in *AWS SDK for Java 2.x API Reference*.

Describe parameters from a DB cluster parameter group

The following code example shows how to describe parameters from an Aurora DB cluster parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

```
                }
            }

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

- For API details, see DescribeDBClusterParameters in *AWS SDK for Java 2.x API Reference*.

Update parameters in a DB cluster parameter group

The following code example shows how to update parameters in an Aurora DB cluster parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see ModifyDBClusterParameterGroup in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with DB clusters

The following code example shows how to:

- Create a custom Aurora DB cluster parameter group and set parameter values.
- Create a DB cluster that uses the parameter group.
- Create a DB instance that contains a database.
- Take a snapshot of the DB cluster, then clean up resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName> <dbSnapshotIdentifier>
 <username> <userPassword>" +
            "Where:\n" +
            "    dbClusterGroupName - The name of the DB cluster parameter group. \n"+
            "    dbParameterGroupFamily - The DB cluster parameter group family name
 (for example, aurora-mysql5.7). \n"+
            "    dbInstanceClusterIdentifier - The instance cluster identifier value.
\n"+
            "    dbInstanceIdentifier - The database instance identifier.\n"+
            "    dbName  The database name.\n"+
            "    dbSnapshotIdentifier - The snapshot identifier.\n"+
            "    username - The database user name.\n" +
            "    userPassword - The database user name password.\n" ;

        if (args.length != 8) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String username = args[6];
        String userPassword = args[7];

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

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

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

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

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

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

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready" );
```

```
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

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

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

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

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

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

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

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

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

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

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

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
```

```
                isDataDel = false ;
                didFind = false;
                int index = 1;
                for (DBInstance instance: instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true ;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the database
ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index ++;
                }
            }

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
```

```
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");

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

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

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


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

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

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

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

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

```
                .maxRecords(20)
                .build();

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

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

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

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

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

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


    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName, String dbClusterIdentifier, String userName,
String password) {
        try {
            CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
                .databaseName(dbName)
```

```
                .dbClusterIdentifier(dbClusterIdentifier)
                .dbClusterParameterGroupName(dbParameterGroupFamily)
                .engine("aurora-mysql")
                .masterUsername(userName)
                .masterUserPassword(password)
                .build();

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

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


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

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

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

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

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

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

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

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

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

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

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

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

```
            }

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

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

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

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

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

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

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

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot

- CreateDBInstance
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

CloudFront examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudFront.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a distribution

The following code example shows how to create a CloudFront distribution.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The following example uses an Amazon Simple Storage Service (Amazon S3) bucket as a content origin.

After creating the distribution, the code creates a CloudFrontWaiter to wait until the distribution is deployed before returning the distribution

.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
```

```
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

    private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

    public static Distribution createDistribution(CloudFrontClient cloudFrontClient,
 S3Client s3Client,
                                                   final String bucketName, final String
 keyGroupId, final String originAccessControlId) {

        final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers().get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region + ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the originId.

        // The service API requires some deprecated methods, such as
 DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .s3OriginConfig(builder4 ->
 builder4.originAccessIdentity(""))
                                        .originAccessControlId(originAccessControlId)))
                        .defaultCacheBehavior(b2 -> b2
                                .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .minTTL(200L)
                                .forwardedValues(b5 -> b5
                                        .cookies(cp -> cp
                                                .forward(ItemSelection.NONE))
                                        .queryString(true))
                                .trustedKeyGroups(b3 -> b3
                                        .quantity(1)
                                        .items(keyGroupId)
                                        .enabled(true))
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)
                                        .cachedMethods(b5 -> b5
                                                .quantity(2)
                                                .items(Method.HEAD, Method.GET))))
                        .cacheBehaviors(b -> b
                                .quantity(1)
                                .items(b2 -> b2
                                        .pathPattern("/index.html")

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                        .targetOriginId(originId)
                                        .trustedKeyGroups(b3 -> b3
                                                .quantity(1)
```

```
                                                .items(keyGroupId)
                                                .enabled(true))
                                        .minTTL(200L)
                                        .forwardedValues(b4 -> b4
                                                .cookies(cp -> cp
                                                        .forward(ItemSelection.NONE))
                                                .queryString(true))
                                        .allowedMethods(b5 -> b5.
                                                quantity(2).
                                                items(Method.HEAD, Method.GET)
                                                .cachedMethods(b6 -> b6
                                                        .quantity(2)
                                                        .items(Method.HEAD,
 Method.GET)))))
                        .enabled(true)
                        .comment("Distribution built with java")
                        .callerReference(Instant.now().toString())
                ));

        final Distribution distribution = createDistResponse.distribution();
        logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
 distribution.domainName(), distribution.id());
        logger.info("Waiting for distribution to be deployed ...");
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
                    cfWaiter.waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id())).matched();
            responseOrException.response().orElseThrow(() -> new
 RuntimeException("Distribution not created"));
            logger.info("Distribution deployed. DomainName: [{}]  Id: [{}]",
 distribution.domainName(), distribution.id());
        }
        return distribution;
    }
}
```

- For API details, see CreateDistribution in *AWS SDK for Java 2.x API Reference*.

Create a function

The following code example shows how to create an Amazon CloudFront function.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createNewFunction(CloudFrontClient cloudFrontClient, String
 functionName, String filePath) {

        try {

            InputStream fileIs =
 CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                .comment("Created by using the CloudFront Java API")
                .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                .build();
```

```
            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .name(functionName)
                .functionCode(functionCode)
                .functionConfig(config)
                .build();

            CreateFunctionResponse response =
 cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

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

- For API details, see CreateFunction in *AWS SDK for Java 2.x API Reference*.

Create a key group

The following code example shows how to create a key group that you can use with signed URLs and signed cookies.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

A key group requires at least one public key that is used to verify signed URLs or cookies.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger = LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.
                        keyGroupConfig(c -> c
                                .items(publicKeyId)
                                .name("JavaKeyGroup"+ UUID.randomUUID())))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- For API details, see CreateKeyGroup in *AWS SDK for Java 2.x API Reference*.

Delete a distribution

The following code example shows how to delete a CloudFront distribution.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The following code example updates a distribution to *disabled*, uses a waiter that waits for the change to be deployed, then deletes the distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

    public static void deleteDistribution(final CloudFrontClient cloudFrontClient,
 final String distributionId) {
        // First, disable the distribution by updating it.
        GetDistributionResponse response = cloudFrontClient.
                getDistribution(b -> b
                        .id(distributionId));
        String etag = response.eTag();
        DistributionConfig distConfig = response.distribution().distributionConfig();

        cloudFrontClient.updateDistribution(builder -> builder
                .id(distributionId)
                .distributionConfig(builder1 -> builder1
                        .cacheBehaviors(distConfig.cacheBehaviors())
                        .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                        .enabled(false)
                        .origins(distConfig.origins())
                        .comment(distConfig.comment())
                        .callerReference(distConfig.callerReference())
                        .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                        .priceClass(distConfig.priceClass())
                        .aliases(distConfig.aliases())
                        .logging(distConfig.logging())
                        .defaultRootObject(distConfig.defaultRootObject())
                        .customErrorResponses(distConfig.customErrorResponses())
                        .httpVersion(distConfig.httpVersion())
                        .isIPV6Enabled(distConfig.isIPV6Enabled())
                        .restrictions(distConfig.restrictions())
                        .viewerCertificate(distConfig.viewerCertificate())
                        .webACLId(distConfig.webACLId())
                        .originGroups(distConfig.originGroups()))
                .ifMatch(etag));

        logger.info("Distribution [{}] is DISABLED, waiting for deployment before
 deleting ...", distributionId);
        GetDistributionResponse distributionResponse;
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
                    cfWaiter.waitUntilDistributionDeployed(builder ->
 builder.id(distributionId)).matched();
            distributionResponse = responseOrException.response().orElseThrow(() -> new
 RuntimeException("Could not disable distribution"));
        }
```

```
        DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient.deleteDistribution(builder -> builder
                .id(distributionId)
                .ifMatch(distributionResponse.eTag()));
        if ( deleteDistributionResponse.sdkHttpResponse().isSuccessful() ){
            logger.info("Distribution [{}] DELETED", distributionId);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - DeleteDistribution
  - UpdateDistribution

Delete signing resources

The following code example shows how to delete resources that are used to gain access to restricted content in an Amazon Simple Storage Service (Amazon S3) bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient, final String originAccessControlId){
        GetOriginAccessControlResponse getResponse =
 cloudFrontClient.getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if ( deleteResponse.sdkHttpResponse().isSuccessful() ){
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
 String keyGroupId){

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
```

```
        DeleteKeyGroupResponse deleteResponse = cloudFrontClient.deleteKeyGroup(builder
 -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if ( deleteResponse.sdkHttpResponse().isSuccessful() ){
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient, final
 String publicKeyId){
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()){
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

Update a distribution

The following code example shows how to update an Amazon CloudFront distribution.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void modDistribution(CloudFrontClient cloudFrontClient, String idVal)
 {

        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                 .id(idVal)
                 .build();

            GetDistributionResponse response =
 cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new  DistributionConfig object and add new values to comment
 and aliases
            DistributionConfig config1 = DistributionConfig.builder()
                .aliases(config.aliases()) // You can pass in new values here
                .comment("New Comment")
                .cacheBehaviors(config.cacheBehaviors())
```

```
                .priceClass(config.priceClass())
                .defaultCacheBehavior(config.defaultCacheBehavior())
                .enabled(config.enabled())
                .callerReference(config.callerReference())
                .logging(config.logging())
                .originGroups(config.originGroups())
                .origins(config.origins())
                .restrictions(config.restrictions())
                .defaultRootObject(config.defaultRootObject())
                .webACLId(config.webACLId())
                .httpVersion(config.httpVersion())
                .viewerCertificate(config.viewerCertificate())
                .customErrorResponses(config.customErrorResponses())
                .build();

            UpdateDistributionRequest updateDistributionRequest =
 UpdateDistributionRequest.builder()
                .distributionConfig(config1)
                .id(disObject.id())
                .ifMatch(response.eTag())
                .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

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

- For API details, see UpdateDistribution in *AWS SDK for Java 2.x API Reference*.

Upload a public key

The following code example shows how to upload a public key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

The following code example reads in a public key and uploads it to Amazon CloudFront.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
 publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
```

```
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse =
 cloudFrontClient.createPublicKey(b -> b.
                    publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- For API details, see CreatePublicKey in *AWS SDK for Java 2.x API Reference*.

Scenarios

Sign URLs and cookies

The following code example shows how to create signed URLs and cookies that allow access to restricted resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use the CannedSignerRequest class to sign URLs or cookies with a *canned* policy.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName, String fileNameToUpload,
                                                                   String
 privateKeyFullPath, String publicKeyId) throws Exception{
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
```

```
    }
}
```

Use the CustomSignerRequest class to sign URLs or cookies with a *custom* policy. The `activeDate` and `ipRange` are optional methods.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName, String fileNameToUpload,
                                                                   String
 privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate)      // Optional.
                //.ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

The following example demonstrates the use of the CloudFrontUtilities class to produce signed cookies and URLs. View this code example on GitHub.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest){
```

```
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest){
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy getCookiesForCannedPolicy(CannedSignerRequest
 cannedSignerRequest){
        CookiesForCannedPolicy cookiesForCannedPolicy =
 cloudFrontUtilities.getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy getCookiesForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy =
 cloudFrontUtilities.getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- For API details, see CloudFrontUtilities in *AWS SDK for Java 2.x API Reference*.

CloudWatch examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello CloudWatch

The following code examples show how to get started using CloudWatch.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage:\n" +
            "  <namespace> \n\n" +
            "Where:\n" +
            "  namespace - The namespace to filter against (for example, AWS/EC2).
 \n" ;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        listMets(cw, namespace);
        cw.close();
    }


    public static void listMets( CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                .namespace(namespace)
                .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                .flatMap(r -> r.metrics().stream())
                .forEach(metrics -> System.out.println(" Retrieved metric is: " +
 metrics.metricName()));

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

- For API details, see ListMetrics in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a dashboard

The following code example shows how to create an Amazon CloudWatch dashboard.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
       try {
           PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
               .dashboardName(dashboardName)
               .dashboardBody(readFileAsString(fileName))
               .build();

           PutDashboardResponse response = cw.putDashboard(dashboardRequest);
           System.out.println(dashboardName +" was successfully created.");
           List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
           if (messages.isEmpty()) {
               System.out.println("There are no messages in the new Dashboard");
           } else {
               for (DashboardValidationMessage message : messages) {
                   System.out.println("Message is: " + message.message());
               }
           }

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

- For API details, see PutDashboard in *AWS SDK for Java 2.x API Reference*.

Create a metric alarm

The following code example shows how to create or update an Amazon CloudWatch alarm and associate it with the specified metric, metric math expression, anomaly detection model, or Metrics Insights query.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createAlarm(CloudWatchClient cw, String fileName) {
```

```
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:"+region+":"+accountId+":"+emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                .alarmActions(alarmActions)
                .alarmDescription("Example metric alarm")
                .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                .threshold(100.00)
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .evaluationPeriods(1)
                .period(10)
                .statistic("Maximum")
                .datapointsToAlarm(1)
                .treatMissingData("ignore")
                .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName +" was successfully created!");
            return alarmName;

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

- For API details, see PutMetricAlarm in *AWS SDK for Java 2.x API Reference*.

Create an anomaly detector

The following code example shows how to create an Amazon CloudWatch anomaly detector.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
 SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
 PutAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric "+customMetricName
+".");

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

- For API details, see PutAnomalyDetector in *AWS SDK for Java 2.x API Reference*.

Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {

        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                .alarmNames(alarmName)
                .build();

            cw.deleteAlarms(request);
            System.out.printf("Successfully deleted alarm %s", alarmName);

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

- For API details, see DeleteAlarms in *AWS SDK for Java 2.x API Reference*.

Delete an anomaly detector

The following code example shows how to delete an Amazon CloudWatch anomaly detector.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- For API details, see DeleteAnomalyDetector in *AWS SDK for Java 2.x API Reference*.

Delete dashboards

The following code example shows how to delete Amazon CloudWatch dashboards.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                .dashboardNames(dashboardName)
                .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteDashboards in *AWS SDK for Java 2.x API Reference*.

Describe alarm history

The following code example shows how to describe an Amazon CloudWatch alarm history.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String alarmName = rootNode.findValue("exampleAlarmName").asText();

           Instant start = Instant.parse(date);
           Instant endDate = Instant.now();
           DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
               .startDate(start)
               .endDate(endDate)
               .alarmName(alarmName)
               .historyItemType(HistoryItemType.ACTION)
               .build();

           DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
           List<AlarmHistoryItem>historyItems = response.alarmHistoryItems();
           if (historyItems.isEmpty()) {
               System.out.println("No alarm history data found for "+alarmName +".");
           } else {
               for (AlarmHistoryItem item: historyItems) {
                   System.out.println("History summary: "+item.historySummary());
                   System.out.println("Time stamp: "+item.timestamp());
               }
           }

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

- For API details, see DescribeAlarmHistory in *AWS SDK for Java 2.x API Reference*.

Describe alarms

The following code example shows how to describe Amazon CloudWatch alarms.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                .alarmTypes(typeList)
                .maxRecords(10)
                .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm: alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " + alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeAlarms in *AWS SDK for Java 2.x API Reference*.

Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .build();

            while (!hasAlarm && retries > 0) {
```

```
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for "+ customMetricName +"
after 10 retries.");
            else
               System.out.println("Alarm state found for "+ customMetricName +".");

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

- For API details, see DescribeAlarmsForMetric in *AWS SDK for Java 2.x API Reference*.

Describe anomaly detectors

The following code example shows how to describe Amazon CloudWatch anomaly detectors.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeAnomalyDetectors(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                .maxResults(10)
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .build();

            DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest) ;
            List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
            for (AnomalyDetector detector: anomalyDetectorList) {
                System.out.println("Metric name:
"+detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: "+detector.stateValue());
            }

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

- For API details, see DescribeAnomalyDetectors in *AWS SDK for Java 2.x API Reference*.

Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void disableActions(CloudWatchClient cw, String alarmName) {

    try {
        DisableAlarmActionsRequest request = DisableAlarmActionsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.disableAlarmActions(request);
        System.out.printf("Successfully disabled actions on alarm %s", alarmName);

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

- For API details, see DisableAlarmActions in *AWS SDK for Java 2.x API Reference*.

Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void enableActions(CloudWatchClient cw, String alarm) {

    try {
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
            .alarmNames(alarm)
            .build();

        cw.enableAlarmActions(request);
        System.out.printf("Successfully enabled actions on alarm %s", alarm);

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

- For API details, see EnableAlarmActions in *AWS SDK for Java 2.x API Reference*.

Get a metric data image

The following code example shows how to get an Amazon CloudWatch metric data image.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
       System.out.println("Getting Image data for custom metric.");
       try {
             String myJSON = "{\n" +
               "  \"title\": \"Example Metric Graph\",\n" +
               "  \"view\": \"timeSeries\",\n" +
               "  \"stacked \": false,\n" +
               "  \"period\": 10,\n" +
               "  \"width\": 1400,\n" +
               "  \"height\": 600,\n" +
               "  \"metrics\": [\n" +
               "    [\n" +
               "      \"AWS/Billing\",\n" +
               "      \"EstimatedCharges\",\n" +
               "      \"Currency\",\n" +
               "      \"USD\"\n" +
               "    ]\n" +
               "  ]\n" +
               "}";

           GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
               .metricWidget(myJSON)
               .build();

           GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
           SdkBytes sdkBytes = response.metricWidgetImage();
           byte[] bytes = sdkBytes.asByteArray();
           File outputFile = new File(fileName);
           try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
               outputStream.write(bytes);
           }

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

- For API details, see GetMetricWidgetImage in *AWS SDK for Java 2.x API Reference*.

Get metric data

The following code example shows how to get Amazon CloudWatch metric data.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName = rootNode.findValue("customMetricName").asText();

           // Set the date.
           Instant nowDate = Instant.now();

           long hours = 1;
           long minutes = 30;
           Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
               ChronoUnit.MINUTES);

           Metric met = Metric.builder()
               .metricName(customMetricName)
               .namespace(customMetricNamespace)
               .build();

           MetricStat metStat = MetricStat.builder()
               .stat("Maximum")
               .period(1)
               .metric(met)
               .build();

           MetricDataQuery dataQUery = MetricDataQuery.builder()
               .metricStat(metStat)
               .id("foo2")
               .returnData(true)
               .build();

           List<MetricDataQuery> dq = new ArrayList<>();
           dq.add(dataQUery);

           GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
               .maxDatapoints(10)
               .scanBy(ScanBy.TIMESTAMP_DESCENDING)
               .startTime(nowDate)
               .endTime(date2)
               .metricDataQueries(dq)
               .build();

           GetMetricDataResponse response = cw.getMetricData(getMetReq);
           List<MetricDataResult> data = response.metricDataResults();
           for (MetricDataResult item : data) {
               System.out.println("The label is " + item.label());
               System.out.println("The status code is " +
item.statusCode().toString());
           }

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

```
    }
```

- For API details, see GetMetricData in *AWS SDK for Java 2.x API Reference*.

Get metric statistics

The following code example shows how to get Amazon CloudWatch metric statistics.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getAndDisplayMetricStatistics( CloudWatchClient cw, String
nameSpace, String metVal, String metricOption, String date, Dimension myDimension) {
       try {
           Instant start = Instant.parse(date);
           Instant endDate = Instant.now();

           GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
               .endTime(endDate)
               .startTime(start)
               .dimensions(myDimension)
               .metricName(metVal)
               .namespace(nameSpace)
               .period(86400)
               .statistics(Statistic.fromValue(metricOption))
               .build();

           GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
           List<Datapoint> data = response.datapoints();
           if (!data.isEmpty()) {
               for (Datapoint datapoint: data) {
                   System.out.println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
               }
           } else {
               System.out.println("The returned data list is empty");
           }

       } catch (CloudWatchException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see GetMetricStatistics in *AWS SDK for Java 2.x API Reference*.

List dashboards

The following code example shows how to list Amazon CloudWatch dashboards.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry ->{
                System.out.println("Dashboard name is: " + entry.dashboardName());
                System.out.println("Dashboard ARN is: " + entry.dashboardArn());
            });

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

- For API details, see ListDashboards in *AWS SDK for Java 2.x API Reference*.

List metrics

The following code example shows how to list the metadata for Amazon CloudWatch metrics. To get data for a metric, use the GetMetricData or GetMetricStatistics actions.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
    String nextToken = null;

    try {
        while(!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

        for (Metric metric : response.metrics()) {
```

```
                System.out.printf("Retrieved metric %s", metric.metricName());
                System.out.println();
            }

            if(response.nextToken() == null) {
                done = true;
            } else {
                nextToken = response.nextToken();
            }
        }

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

- For API details, see ListMetrics in *AWS SDK for Java 2.x API Reference*.

Put data into a metric

The following code example shows how to publish metric data points to Amazon CloudWatch.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();

            // Set an Instant object.
            String time =
ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                .metricName(customMetricName)
                .unit(StandardUnit.NONE)
                .value(1001.00)
                .timestamp(instant)
                .build();

            MetricDatum datum2 = MetricDatum.builder()
                .metricName(customMetricName)
                .unit(StandardUnit.NONE)
                .value(1002.00)
                .timestamp(instant)
                .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);
```

```
            PutMetricDataRequest request = PutMetricDataRequest.builder()
                .namespace(customMetricNamespace)
                .metricData(metricDataList)
                .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric "
 +customMetricName);

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

- For API details, see PutMetricData in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with metrics, dashboards, and alarms

The following code example shows how to:

- List CloudWatch namespaces and metrics.
- Get statistics for a metric and for estimated billing.
- Create and update a dashboard.
- Create and add data to a metric.
- Create and trigger an alarm, then view alarm history.
- Add an anomaly detector.
- Get a metric image, then clean up resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing alerts
 are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
```

```
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws IOException {
        final String usage = "\n" +
            "Usage:\n" +
            "  <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage>  \n\n" +
            "Where:\n" +
            "  myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.) \n" +
            "  costDateWeek - The start date to use to get AWS/Billinget statistics.
 (For example, 2023-01-11T18:35:24.00Z.) \n" +
            "  dashboardName - The name of the dashboard to create. \n" +
            "  dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.) \n" +
            "  dashboardAdd - The location of a JSON file to use to update a dashboard.
 (See Readme file.) \n" +
            "  settings - The location of a JSON file from which various values are
 read. (See Readme file.) \n" +
            "  metricImage - The location of a BMP file that is used to create a graph.
 \n" ;

       if (args.length != 7) {
           System.out.println(usage);
           System.exit(1);
       }

        Region region = Region.US_EAST_1;
        String myDate = args[0];
        String costDateWeek = args[1];
        String dashboardName = args[2];
        String dashboardJson = args[3];
        String dashboardAdd = args[4];
        String settings = args[5];
        String metricImage = args[6];


        Double dataPoint = Double.parseDouble("10.0");
        Scanner sc = new Scanner(System.in);
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon CloudWatch example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. List at least five available unique namespaces from
 Amazon CloudWatch. Select one from the list.");
        ArrayList<String> list = listNameSpaces(cw);
```

```
        for (int z=0; z<5; z++) {
            int index = z+1;
            System.out.println("    " +index +". " +list.get(z));
        }

        String selectedNamespace = "";
        String selectedMetrics = "";
        int num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5){
            selectedNamespace = list.get(num-1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected "+selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List available metrics within the selected namespace and
select one from the list.");
        ArrayList<String> metList = listMets(cw, selectedNamespace);
        for (int z=0; z<5; z++) {
            int index = z+1;
            System.out.println("    " +index +". " +metList.get(z));
        }
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5){
            selectedMetrics = metList.get(num-1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected "+selectedMetrics);
        Dimension myDimension = getSpecificMet( cw, selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get statistics for the selected metric over the last
day.");
        String metricOption="";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t=0; t<5; t++){
            System.out.println("    " +(t+1) +". "+statTypes.get(t));
        }
        System.out.println("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5){
            metricOption = statTypes.get(num-1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected "+metricOption);
        getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get CloudWatch estimated billing for the last week.");
```

```
        getMetricStatistics(cw, costDateWeek);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new CloudWatch dashboard with metrics.");
        createDashboardWithMetrics(cw, dashboardName, dashboardJson);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List dashboards using a paginator.");
        listDashboards(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a new custom metric by adding data to it.");
        createNewCustomMetric(cw, dataPoint);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Add an additional metric to the dashboard.");
        addMetricToDashboard(cw, dashboardAdd, dashboardName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Create an alarm for the custom metric.");
        String alarmName = createAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Describe ten current alarms.");
        describeAlarms(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Get current data for new custom metric.");
        getCustomMetricData(cw,settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Push data into the custom metric to trigger the
alarm.");
        addMetricDataForAlarm(cw, settings) ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        checkForMetricAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Get alarm history for the new alarm.");
        getAlarmHistory(cw, settings, myDate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Add an anomaly detector for the custom metric.");
        addAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);

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

```
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                .alarmNames(alarmName)
                .build();

            cw.deleteAlarms(request);
            System.out.println("Successfully deleted alarm " +alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
```

```
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                .dashboardNames(dashboardName)
                .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
        System.out.println("Getting Image data for custom metric.");
        try {
              String myJSON = "{\n" +
                "  \"title\": \"Example Metric Graph\",\n" +
                "  \"view\": \"timeSeries\",\n" +
                "  \"stacked \": false,\n" +
                "  \"period\": 10,\n" +
                "  \"width\": 1400,\n" +
                "  \"height\": 600,\n" +
                "  \"metrics\": [\n" +
                "    [\n" +
                "      \"AWS/Billing\",\n" +
                "      \"EstimatedCharges\",\n" +
                "      \"Currency\",\n" +
                "      \"USD\"\n" +
                "    ]\n" +
                "  ]\n" +
                "}";

            GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
                .metricWidget(myJSON)
                .build();

            GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAnomalyDetectors(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                .maxResults(10)
```

```
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .build();

            DescribeAnomalyDetectorsResponse response =
 cw.describeAnomalyDetectors(detectorsRequest) ;
            List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
            for (AnomalyDetector detector: anomalyDetectorList) {
                System.out.println("Metric name:
 "+detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: "+detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
 SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
 PutAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric "+customMetricName
+".");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
 date) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();

            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
 DescribeAlarmHistoryRequest.builder()
                .startDate(start)
                .endDate(endDate)
```

```
                .alarmName(alarmName)
                .historyItemType(HistoryItemType.ACTION)
                .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem>historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for "+alarmName +".");
            } else {
                for (AlarmHistoryItem item: historyItems) {
                    System.out.println("History summary: "+item.historySummary());
                    System.out.println("Time stamp: "+item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for "+ customMetricName +"
after 10 retries.");
            else
               System.out.println("Alarm state found for "+ customMetricName +".");

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

    public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();

            // Set an Instant object.
            String time =
ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                .metricName(customMetricName)
                .unit(StandardUnit.NONE)
                .value(1001.00)
                .timestamp(instant)
                .build();

            MetricDatum datum2 = MetricDatum.builder()
                .metricName(customMetricName)
                .unit(StandardUnit.NONE)
                .value(1002.00)
                .timestamp(instant)
                .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                .namespace(customMetricNamespace)
                .metricData(metricDataList)
                .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric "
+customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName = rootNode.findValue("customMetricName").asText();

            // Set the date.
            Instant nowDate = Instant.now();

            long hours = 1;
            long minutes = 30;
            Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                ChronoUnit.MINUTES);

            Metric met = Metric.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .build();

            MetricStat metStat = MetricStat.builder()
```

```
                .stat("Maximum")
                .period(1)
                .metric(met)
                .build();

            MetricDataQuery dataQUery = MetricDataQuery.builder()
                .metricStat(metStat)
                .id("foo2")
                .returnData(true)
                .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);

            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                .maxDatapoints(10)
                .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                .startTime(nowDate)
                .endTime(date2)
                .metricDataQueries(dq)
                .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                .alarmTypes(typeList)
                .maxRecords(10)
                .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm: alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " + alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
            String customMetricName = rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:"+region+":"+accountId+":"+emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                .alarmActions(alarmActions)
                .alarmDescription("Example metric alarm")
                .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                .threshold(100.00)
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .evaluationPeriods(1)
                .period(10)
                .statistic("Maximum")
                .datapointsToAlarm(1)
                .treatMissingData("ignore")
                .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName +" was successfully created!");
            return alarmName;

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

    public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                .dashboardName(dashboardName)
                .dashboardBody(readFileAsString(fileName))
                .build();

            cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName +" was successfully updated.");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint) {
        try {
            Dimension dimension = Dimension.builder()
                .name("UNIQUE_PAGES")
                .value("URLS")
                .build();

            // Set an Instant object.
            String time =
ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
```

```
                .metricName("PAGES_VISITED")
                .unit(StandardUnit.NONE)
                .value(dataPoint)
                .timestamp(instant)
                .dimensions(dimension)
                .build();

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                .namespace("SITE/TRAFFIC")
                .metricData(datum)
                .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric PAGES_VISITED");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                .flatMap(r -> r.dashboardEntries().stream())
                .forEach(entry ->{
                    System.out.println("Dashboard name is: " + entry.dashboardName());
                    System.out.println("Dashboard ARN is: " + entry.dashboardArn());
                });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createDashboardWithMetrics(CloudWatchClient cw, String
 dashboardName, String fileName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                .dashboardName(dashboardName)
                .dashboardBody(readFileAsString(fileName))
                .build();

            PutDashboardResponse response = cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName +" was successfully created.");
            List<DashboardValidationMessage> messages =
 response.dashboardValidationMessages();
            if (messages.isEmpty()) {
                System.out.println("There are no messages in the new Dashboard");
            } else {
                for (DashboardValidationMessage message : messages) {
                    System.out.println("Message is: " + message.message());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
```

```
   public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek) {
       try {
           Instant start = Instant.parse(costDateWeek);
           Instant endDate = Instant.now();
           Dimension dimension = Dimension.builder()
               .name("Currency")
               .value("USD")
               .build();

           List<Dimension> dimensionList = new ArrayList<>();
           dimensionList.add(dimension);
           GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
               .metricName("EstimatedCharges")
               .namespace("AWS/Billing")
               .dimensions(dimensionList)
               .statistics(Statistic.MAXIMUM)
               .startTime(start)
               .endTime(endDate)
               .period(86400)
               .build();

           GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
           List<Datapoint> data = response.datapoints();
           if (!data.isEmpty()) {
               for (Datapoint datapoint: data) {
                   System.out.println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
               }
           } else {
               System.out.println("The returned data list is empty");
           }

       } catch (CloudWatchException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void getAndDisplayMetricStatistics( CloudWatchClient cw, String
nameSpace, String metVal, String metricOption, String date, Dimension myDimension) {
       try {
           Instant start = Instant.parse(date);
           Instant endDate = Instant.now();

           GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
               .endTime(endDate)
               .startTime(start)
               .dimensions(myDimension)
               .metricName(metVal)
               .namespace(nameSpace)
               .period(86400)
               .statistics(Statistic.fromValue(metricOption))
               .build();

           GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
           List<Datapoint> data = response.datapoints();
           if (!data.isEmpty()) {
               for (Datapoint datapoint: data) {
                   System.out.println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
               }
```

```
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static Dimension getSpecificMet( CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                .namespace(namespace)
                .build();

            ListMetricsResponse response = cw.listMetrics(request);
            List<Metric> myList = response.metrics();
            Metric metric = myList.get(0);
            return metric.dimensions().get(0);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static ArrayList<String> listMets( CloudWatchClient cw, String namespace) {
        try {
            ArrayList<String> metList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                .namespace(namespace)
                .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                .flatMap(r -> r.metrics().stream())
                .forEach(metrics -> metList.add(metrics.metricName()));

            return metList;

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

   public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
       try {
           ArrayList<String> nameSpaceList = new ArrayList<>();
           ListMetricsRequest request = ListMetricsRequest.builder()
               .build();

           ListMetricsIterable listRes = cw.listMetricsPaginator(request);
           listRes.stream()
               .flatMap(r -> r.metrics().stream())
               .forEach(metrics -> {
                   String data = metrics.namespace();
                   if(!nameSpaceList.contains(data)) {
                       nameSpaceList.add(data);
                   }
               }) ;

           return nameSpaceList;
```

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

CloudWatch Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch Events.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Adding a target

The following code example shows how to add a target to an Amazon CloudWatch Events event.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

       try {
           Target target = Target.builder()
               .arn(functionArn)
               .id(targetId)
               .build();

           PutTargetsRequest request = PutTargetsRequest.builder()
               .targets(target)
               .rule(ruleName)
               .build();

           cwe.putTargets(request);
           System.out.printf(
               "Successfully created CloudWatch events target for rule %s",
               ruleName);

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

- For API details, see PutTargets in *AWS SDK for Java 2.x API Reference*.

Create a scheduled rule

The following code example shows how to create an Amazon CloudWatch Events scheduled rule.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
roleArn) {

       try {
           PutRuleRequest request = PutRuleRequest.builder()
               .name(ruleName)
               .roleArn(roleArn)
               .scheduleExpression("rate(5 minutes)")
               .state(RuleState.ENABLED)
               .build();

           PutRuleResponse response = cwe.putRule(request);
           System.out.printf(
                   "Successfully created CloudWatch events rule %s with arn %s",
                   roleArn, response.ruleArn());

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

- For API details, see PutRule in *AWS SDK for Java 2.x API Reference*.

Send events

The following code example shows how to send Amazon CloudWatch Events events.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn ) {
    try {

        final String EVENT_DETAILS =
            "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
            .detail(EVENT_DETAILS)
            .detailType("sampleSubmitted")
            .resources(resourceArn)
            .source("aws-sdk-java-cloudwatch-example")
            .build();

        PutEventsRequest request = PutEventsRequest.builder()
            .entries(requestEntry)
            .build();

        cwe.putEvents(request);
        System.out.println("Successfully put CloudWatch event");

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

- For API details, see PutEvents in *AWS SDK for Java 2.x API Reference*.

CloudWatch Logs examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with CloudWatch Logs.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a subscription filter

The following code example shows how to create an Amazon CloudWatch Logs subscription filter.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void putSubFilters(CloudWatchLogsClient cwl,
                                    String filter,
                                    String pattern,
                                    String logGroup,
                                    String functionArn) {

       try {
           PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
               .filterName(filter)
               .filterPattern(pattern)
               .logGroupName(logGroup)
               .destinationArn(functionArn)
               .build();

           cwl.putSubscriptionFilter(request);
           System.out.printf(
                   "Successfully created CloudWatch logs subscription filter %s",
                   filter);

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

- For API details, see PutSubscriptionFilter in *AWS SDK for Java 2.x API Reference*.

Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteSubFilter(CloudWatchLogsClient logs, String filter, String
logGroup) {

       try {
           DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
               .filterName(filter)
               .logGroupName(logGroup)
               .build();
```

```
            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription filter
%s", filter);

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

- For API details, see DeleteSubscriptionFilter in *AWS SDK for Java 2.x API Reference*.

Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {

        try {
            boolean done = false;
            String newToken = null;

            while(!done) {
                DescribeSubscriptionFiltersResponse response;
                if (newToken == null) {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                        .logGroupName(logGroup)
                        .limit(1).build();

                    response = logs.describeSubscriptionFilters(request);
                } else {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                        .nextToken(newToken)
                        .logGroupName(logGroup)
                        .limit(1).build();
                    response = logs.describeSubscriptionFilters(request);
                }

                for(SubscriptionFilter filter : response.subscriptionFilters()) {
                    System.out.printf("Retrieved filter with name %s, " + "pattern %s "
+ "and destination arn %s",
                        filter.filterName(),
                        filter.filterPattern(),
                        filter.destinationArn());
                }

                if(response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }
```

```
        } catch (CloudWatchException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
        }
        System.out.printf("Done");
    }
```

- For API details, see DescribeSubscriptionFilters in *AWS SDK for Java 2.x API Reference*.

Amazon Cognito Identity Provider examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Cognito Identity Provider.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {
       try {
           ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
               .clientId(clientId)
               .confirmationCode(code)
               .username(userName)
               .build();

           identityProviderClient.confirmSignUp(signUpRequest);
           System.out.println(userName +" was confirmed");

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

- For API details, see ConfirmSignUp in *AWS SDK for Java 2.x API Reference*.

Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
       AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
           .session(session)
           .build();

       AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
   }
```

- For API details, see AssociateSoftwareToken in *AWS SDK for Java 2.x API Reference*.

Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {
       try {
           AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
               .username(userName)
               .userPoolId(poolId)
               .build();

           AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
           System.out.println("User status "+response.userStatusAsString());

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

- For API details, see AdminGetUser in *AWS SDK for Java 2.x API Reference*.

List users

The following code example shows how to list Amazon Cognito users.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listAllUsers(CognitoIdentityProviderClient cognitoClient, String
userPoolId ) {
       try {
           ListUsersRequest usersRequest = ListUsersRequest.builder()
               .userPoolId(userPoolId)
               .build();

           ListUsersResponse response = cognitoClient.listUsers(usersRequest);
           response.users().forEach(user -> {
               System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created " + user.userCreateDate() );
           });

       } catch (CognitoIdentityProviderException e){
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   // Shows how to list users by using a filter.
   public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId ) {

       try {
           String filter = "email = \"tblue@noserver.com\"";
           ListUsersRequest usersRequest = ListUsersRequest.builder()
               .userPoolId(userPoolId)
               .filter(filter)
               .build();

           ListUsersResponse response = cognitoClient.listUsers(usersRequest);
           response.users().forEach(user -> {
               System.out.println("User with filter applied " + user.username() + "
Status " + user.userStatus() + " Created " + user.userCreateDate() );
           });

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

- For API details, see ListUsers in *AWS SDK for Java 2.x API Reference*.

Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
               .clientId(clientId)
               .username(userName)
               .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

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

- For API details, see ResendConfirmationCode in *AWS SDK for Java 2.x API Reference*.

Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Respond to an authentication challenge.
   public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {
       System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
       Map<String, String> challengeResponses = new HashMap<>();

       challengeResponses.put("USERNAME", userName);
       challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
           .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
           .clientId(clientId)
           .challengeResponses(challengeResponses)
           .session(session)
           .build();

       AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.adminRespondToAuthChallenge(respondToAuthChallengeRequest);
       System.out.println("respondToAuthChallengeResult.getAuthenticationResult()" +
respondToAuthChallengeResult.authenticationResult());
```

```
    }
```

- For API details, see AdminRespondToAuthChallenge in *AWS SDK for Java 2.x API Reference*.

Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
            .name("email")
            .value(email)
            .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                .userAttributes(userAttrsList)
                .username(userName)
                .clientId(clientId)
                .password(password)
                .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

- For API details, see SignUp in *AWS SDK for Java 2.x API Reference*.

Start authentication with administrator credentials

The following code example shows how to start authentication with Amazon Cognito and administrator credentials.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static AdminInitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password, String
userPoolId) {
```

```
        try {
            Map<String,String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest = AdminInitiateAuthRequest.builder()
                .clientId(clientId)
                .userPoolId(userPoolId)
                .authParameters(authParameters)
                .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName() );
            return response;

        } catch(CognitoIdentityProviderException e) {
               System.err.println(e.awsErrorDetails().errorMessage());
               System.exit(1);
        }

        return null;
    }
```

- For API details, see AdminInitiateAuth in *AWS SDK for Java 2.x API Reference*.

Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient identityProviderClient,
String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                .userCode(code)
                .session(session)
                .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is "
+verifyResponse.statusAsString());

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

- For API details, see VerifySoftwareToken in *AWS SDK for Java 2.x API Reference*.

Scenarios

Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up and confirm a user with a username, password, and email address.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 to set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted to
 submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <clientId> <poolId>\n\n" +
            "Where:\n" +
            "    clientId - The app client Id value that you can get from the AWS CDK
 script.\n\n" +
            "    poolId - The pool Id that you can get from the AWS CDK script. \n\n" ;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out.println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse = initiateAuth(identityProviderClient,
clientId, userName, password, poolId) ;
        String mySession = authResponse.session() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate a
TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
```

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

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 = initiateAuth(identityProviderClient,
clientId, userName, password, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
            .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
            .clientId(clientId)
            .challengeResponses(challengeResponses)
            .session(session)
            .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()" +
respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient identityProviderClient,
String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                .userCode(code)
                .session(session)
```

```
                .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is "
+verifyResponse.statusAsString());

        } catch(CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static AdminInitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password, String
userPoolId) {
        try {
            Map<String,String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest = AdminInitiateAuthRequest.builder()
                .clientId(clientId)
                .userPoolId(userPoolId)
                .authParameters(authParameters)
                .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName() );
            return response;

        } catch(CognitoIdentityProviderException e) {
               System.err.println(e.awsErrorDetails().errorMessage());
               System.exit(1);
        }

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
            .session(session)
            .build();

        AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                .clientId(clientId)
                .confirmationCode(code)
                .username(userName)
                .build();

            identityProviderClient.confirmSignUp(signUpRequest);
```

```
            System.out.println(userName +" was confirmed");

        } catch(CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                .clientId(clientId)
                .username(userName)
                .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

        } catch(CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
            .name("email")
            .value(email)
            .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                .userAttributes(userAttrsList)
                .username(userName)
                .clientId(clientId)
                .password(password)
                .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch(CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }


    public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                .username(userName)
                .userPoolId(poolId)
                .build();

            AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status "+response.userStatusAsString());
```

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Amazon Comprehend examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Comprehend.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a document classifier

The following code example shows how to create an Amazon Comprehend document classifier.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createDocumentClassifier(ComprehendClient comClient, String
 dataAccessRoleArn, String s3Uri, String documentClassifierName){
```

```
        try {
            DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
                .s3Uri(s3Uri)
                .build();

            CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
                .documentClassifierName(documentClassifierName)
                .dataAccessRoleArn(dataAccessRoleArn)
                .languageCode("en")
                .inputDataConfig(config)
                .build();

            CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient.createDocumentClassifier(createDocumentClassifierRequest);
            String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " + documentClassifierArn);

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

- For API details, see CreateDocumentClassifier in *AWS SDK for Java 2.x API Reference*.

Detect entities in a document

The following code example shows how to detect entities in a document with Amazon Comprehend.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void detectAllEntities(ComprehendClient comClient,String text ) {

       try {
           DetectEntitiesRequest detectEntitiesRequest =
DetectEntitiesRequest.builder()
               .text(text)
               .languageCode("en")
               .build();

           DetectEntitiesResponse detectEntitiesResult =
comClient.detectEntities(detectEntitiesRequest);
           List<Entity> entList = detectEntitiesResult.entities();
           for (Entity entity : entList) {
               System.out.println("Entity text is " + entity.text());
           }

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

- For API details, see DetectEntities in *AWS SDK for Java 2.x API Reference*.

Detect key phrases in a document

The following code example shows how to detect key phrases in a document with Amazon Comprehend.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void detectAllKeyPhrases(ComprehendClient comClient, String text) {

       try {
           DetectKeyPhrasesRequest detectKeyPhrasesRequest =
DetectKeyPhrasesRequest.builder()
               .text(text)
               .languageCode("en")
               .build();

           DetectKeyPhrasesResponse detectKeyPhrasesResult =
comClient.detectKeyPhrases(detectKeyPhrasesRequest);
           List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
           for (KeyPhrase keyPhrase : phraseList) {
               System.out.println("Key phrase text is " + keyPhrase.text());
           }

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

- For API details, see DetectKeyPhrases in *AWS SDK for Java 2.x API Reference*.

Detect syntactical elements of a document

The following code example shows how to detect syntactial elements of a document with Amazon Comprehend.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void detectAllSyntax(ComprehendClient comClient, String text){

       try {
           DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
               .text(text)
               .languageCode("en")
               .build();

           DetectSyntaxResponse detectSyntaxResult =
comClient.detectSyntax(detectSyntaxRequest);
           List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
           for (SyntaxToken token : syntaxTokens) {
```

```
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
token.partOfSpeech().tagAsString());
            }

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

- For API details, see DetectSyntax in *AWS SDK for Java 2.x API Reference*.

Detect the dominant language in a document

The following code example shows how to detect the dominant language in a document with Amazon Comprehend.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void detectTheDominantLanguage(ComprehendClient comClient, String
text){

        try {
            DetectDominantLanguageRequest request =
DetectDominantLanguageRequest.builder()
                .text(text)
                .build();

            DetectDominantLanguageResponse resp =
comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

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

- For API details, see DetectDominantLanguage in *AWS SDK for Java 2.x API Reference*.

Detect the sentiment of a document

The following code example shows how to detect the sentiment of a document with Amazon Comprehend.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void detectSentiments(ComprehendClient comClient, String text){

       try {
           DetectSentimentRequest detectSentimentRequest =
DetectSentimentRequest.builder()
               .text(text)
               .languageCode("en")
               .build();

           DetectSentimentResponse detectSentimentResult =
comClient.detectSentiment(detectSentimentRequest);
           System.out.println("The Neutral value is "
+detectSentimentResult.sentimentScore().neutral() );

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

- For API details, see DetectSentiment in *AWS SDK for Java 2.x API Reference*.

DynamoDB examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with DynamoDB.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a table

The following code example shows how to create a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createTable(DynamoDbClient ddb, String tableName, String key)
{
       DynamoDbWaiter dbWaiter = ddb.waiter();
       CreateTableRequest request = CreateTableRequest.builder()
           .attributeDefinitions(AttributeDefinition.builder()
```

```
                .attributeName(key)
                .attributeType(ScalarAttributeType.S)
                .build())
            .keySchema(KeySchemaElement.builder()
                .attributeName(key)
                .keyType(KeyType.HASH)
                .build())
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .tableName(tableName)
            .build();

        String newTable ="";
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            newTable = response.tableDescription().tableName();
            return newTable;

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

- For API details, see CreateTable in *AWS SDK for Java 2.x API Reference*.

Delete a table

The following code example shows how to delete a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName +" was successfully deleted!");
```

```
    }
```

- For API details, see DeleteTable in *AWS SDK for Java 2.x API Reference*.

Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
       HashMap<String,AttributeValue> keyToGet = new HashMap<>();
       keyToGet.put(key, AttributeValue.builder()
           .s(keyVal)
           .build());

       DeleteItemRequest deleteReq = DeleteItemRequest.builder()
           .tableName(tableName)
           .key(keyToGet)
           .build();

       try {
           ddb.deleteItem(deleteReq);
       } catch (DynamoDbException e) {
          System.err.println(e.getMessage());
          System.exit(1);
       }
   }
```

- For API details, see DeleteItem in *AWS SDK for Java 2.x API Reference*.

Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Gets an item from a table by using the DynamoDbClient.

```
   public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal, String partitionAlias) {

       // Set up an alias for the partition key name in case it's a reserved word.
       HashMap<String,String> attrNameAlias = new HashMap<String,String>();
       attrNameAlias.put(partitionAlias, partitionKeyName);

       // Set up mapping of the partition name with the value.
       HashMap<String, AttributeValue> attrValues = new HashMap<>();
```

```
        attrValues.put(":"+partitionKeyName, AttributeValue.builder()
            .s(partitionKeyVal)
            .build());

        QueryRequest queryReq = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
            .expressionAttributeNames(attrNameAlias)
            .expressionAttributeValues(attrValues)
            .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();

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

- For API details, see GetItem in *AWS SDK for Java 2.x API Reference*.

Get information about a table

The following code example shows how to get information about a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeDymamoDBTable(DynamoDbClient ddb,String tableName ) {

        DescribeTableRequest request = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
                System.out.format("Table name  : %s\n", tableInfo.tableName());
                System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
                System.out.format("Status      : %s\n", tableInfo.tableStatus());
                System.out.format("Item count  : %d\n",
tableInfo.itemCount().longValue());
                System.out.format("Size (bytes): %d\n",
tableInfo.tableSizeBytes().longValue());

                ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits().longValue());
                System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits().longValue());

                List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
```

```
                System.out.println("Attributes");

                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
```

- For API details, see DescribeTable in *AWS SDK for Java 2.x API Reference*.

List tables

The following code example shows how to list DynamoDB tables.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllTables(DynamoDbClient ddb){

        boolean moreTables = true;
        String lastName = null;

        while(moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
```

```
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
```

- For API details, see ListTables in *AWS SDK for Java 2.x API Reference*.

Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Puts an item into a table using DynamoDbClient.

```
    public static void putItemInTable(DynamoDbClient ddb,
                                      String tableName,
                                      String key,
                                      String keyVal,
                                      String albumTitle,
                                      String albumTitleValue,
                                      String awards,
                                      String awardVal,
                                      String songTitle,
                                      String songTitleVal){

        HashMap<String,AttributeValue> itemValues = new HashMap<>();
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle,
 AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
            .tableName(tableName)
            .item(itemValues)
            .build();

        try {
            PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName +" was successfully updated. The request id is
 "+response.responseMetadata().requestId());

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be found.
\n", tableName);
            System.err.println("Be sure that it exists and that you've typed its name
 correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutItem in *AWS SDK for Java 2.x API Reference*.

Query a table

The following code example shows how to query a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Queries a table by using DynamoDbClient.

```
   public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal, String partitionAlias) {

       // Set up an alias for the partition key name in case it's a reserved word.
       HashMap<String,String> attrNameAlias = new HashMap<String,String>();
       attrNameAlias.put(partitionAlias, partitionKeyName);

       // Set up mapping of the partition name with the value.
       HashMap<String, AttributeValue> attrValues = new HashMap<>();

       attrValues.put(":"+partitionKeyName, AttributeValue.builder()
           .s(partitionKeyVal)
           .build());

       QueryRequest queryReq = QueryRequest.builder()
           .tableName(tableName)
           .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
           .expressionAttributeNames(attrNameAlias)
           .expressionAttributeValues(attrValues)
           .build();

       try {
           QueryResponse response = ddb.query(queryReq);
           return response.count();

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

Queries a table by using `DynamoDbClient` and a secondary index.

```
   public static void queryIndex(DynamoDbClient ddb, String tableName) {

       try {
           Map<String, String> expressionAttributesNames = new HashMap<>();
           expressionAttributesNames.put("#year", "year");
           Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
           expressionAttributeValues.put(":yearValue",
AttributeValue.builder().n("2013").build());

           QueryRequest request = QueryRequest.builder()
               .tableName(tableName)
               .indexName("year-index")
               .keyConditionExpression("#year = :yearValue")
               .expressionAttributeNames(expressionAttributesNames)
               .expressionAttributeValues(expressionAttributeValues)
               .build();
```

```
            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see Query in *AWS SDK for Java 2.x API Reference*.

Scan a table

The following code example shows how to scan a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Scans an Amazon DynamoDB table using DynamoDbClient.

```
    public static void scanItems( DynamoDbClient ddb,String tableName ) {

        try {
            ScanRequest scanRequest = ScanRequest.builder()
                .tableName(tableName)
                .build();

            ScanResponse response = ddb.scan(scanRequest);
            for (Map<String, AttributeValue> item : response.items()) {
                Set<String> keys = item.keySet();
                for (String key : keys) {
                    System.out.println ("The key name is "+key +"\n" );
                    System.out.println("The value is "+item.get(key).s());
                }
            }

        } catch (DynamoDbException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
```

- For API details, see Scan in *AWS SDK for Java 2.x API Reference*.

Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Updates an item in a table using DynamoDbClient.

```
    public static void updateTableItem(DynamoDbClient ddb,
                                       String tableName,
                                       String key,
                                       String keyVal,
                                       String name,
                                       String updateVal){

        HashMap<String,AttributeValue> itemKey = new HashMap<>();
        itemKey.put(key, AttributeValue.builder()
            .s(keyVal)
            .build());

        HashMap<String,AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put(name, AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s(updateVal).build())
            .action(AttributeAction.PUT)
            .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(itemKey)
            .attributeUpdates(updatedValues)
            .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("The Amazon DynamoDB table was updated!");
    }
```

- For API details, see UpdateItem in *AWS SDK for Java 2.x API Reference*.

Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Inserts many items into a table by using the enhanced client.

```
    public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {

        try {
            DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
            DynamoDbTable<Music> musicMappedTable = enhancedClient.table("Music",
TableSchema.fromBean(Music.class));
            LocalDate localDate = LocalDate.parse("2020-04-07");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            Customer record2 = new Customer();
```

```
            record2.setCustName("Fred Pink");
            record2.setId("id110");
            record2.setEmail("fredp@noserver.com");
            record2.setRegistrationDate(instant) ;

            Customer record3 = new Customer();
            record3.setCustName("Susan Pink");
            record3.setId("id120");
            record3.setEmail("spink@noserver.com");
            record3.setRegistrationDate(instant) ;

            Customer record4 = new Customer();
            record4.setCustName("Jerry orange");
            record4.setId("id101");
            record4.setEmail("jorange@noserver.com");
            record4.setRegistrationDate(instant) ;


            BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest =
BatchWriteItemEnhancedRequest.builder()
                    .writeBatches(
                            WriteBatch.builder(Customer.class)          // add items to
the Customer table
                                    .mappedTableResource(customerMappedTable)
                                    .addPutItem(builder -> builder.item(record2))
                                    .addPutItem(builder -> builder.item(record3))
                                    .addPutItem(builder -> builder.item(record4))
                                    .build(),
                            WriteBatch.builder(Music.class)             // delete an
item from the Music table
                                    .mappedTableResource(musicMappedTable)
                                    .addDeleteItem(builder -> builder.key(
                                            Key.builder().partitionValue("Famous
Band").build()))
                                    .build())
                    .build();


            // Add three items to the Customer table and delete one item from the Music
table
            enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);

            System.out.println("done");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see BatchWriteItem in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.

- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table, then delete the table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a DynamoDB table.

```
    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
```

```
            String newTable = response.tableDescription().tableName();
            System.out.println("The " +newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Create a helper function to download and extract the sample JSON file.

```
    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String fileName)
throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0 ;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break ;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }
```

Get an item from a table.

```
    public static void getItem(DynamoDbClient ddb) {

        HashMap<String,AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
```

```
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
             System.err.println(e.getMessage());
             System.exit(1);
        }
    }
```

Full example.

```
/**
 *  Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  This Java example performs these tasks:
 *
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the Enhanced
 client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws IOException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fileName>\n\n" +
            "Where:\n" +
            "    fileName - The path to the moviedata.json file that you can download
 from the Amazon DynamoDB Developer Guide.\n" ;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
```

```
        String fileName = args[0];
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Creating an Amazon DynamoDB table named Movies with a
key named year and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb) ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
```

```
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " +newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Query the table.
    public static void queryTable(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            QueryConditional queryConditional = QueryConditional
                .keyEqualTo(Key.builder()
                .partitionValue(2013)
                .build());

            // Get items in the table and write out the ID value.
```

```
            Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
            String result="";

            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The title of the movie is "+rec.getTitle());
                System.out.println("The movie information  is "+rec.getInfo());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
            }
        }

        // Scan the table.
        public static void scanMovies(DynamoDbClient ddb, String tableName) {
            System.out.println("******* Scanning all movies.\n");
            try{
                DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
                        .dynamoDbClient(ddb)
                        .build();

                DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
                Iterator<Movies> results = custTable.scan().items().iterator();
                while (results.hasNext()) {
                    Movies rec = results.next();
                    System.out.println("The movie title is "+rec.getTitle());
                    System.out.println("The movie year is " +rec.getYear());
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String fileName)
throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0 ;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break ;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
```

```
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName){
        HashMap<String,AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String,AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack\"]").build())
            .action(AttributeAction.PUT)
            .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(itemKey)
            .attributeUpdates(updatedValues)
            .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Item was updated!");
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName +" was successfully deleted!");
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));
```

```
            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String,AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
             System.err.println(e.getMessage());
             System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query

- Scan
- UpdateItem

Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class ScenarioPartiQLBatch {

    public static void main(String [] args) throws IOException {

        String tableName = "MoviesPartiQBatch";
        ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .credentialsProvider(credentialsProvider)
            .region(region)
            .build();

        System.out.println("******* Creating an Amazon DynamoDB table named "+tableName
 +" with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("******* Adding multiple records into the "+ tableName +"
 table using a batch command.");
        putRecordBatch(ddb);

        System.out.println("******* Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("******* Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);

        System.out.println("******* Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());
```

```
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " +newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie 1")
                .build();

            AttributeValue att3 = AttributeValue.builder()
```

```
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parameters)
                .build();

            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("My Movie 2")
                .build();

            AttributeValue attMovie2B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie2.add(attMovie2);
            parametersMovie2.add(attMovie2A);
            parametersMovie2.add(attMovie2B);

            BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie2)
                .build();

            // Set data for Movie 3.
            List<AttributeValue> parametersMovie3 = new ArrayList<>();
            AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie3A = AttributeValue.builder()
                .s("My Movie 3")
                .build();

            AttributeValue attMovie3B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie3.add(attMovie3);
            parametersMovie3.add(attMovie3A);
            parametersMovie3.add(attMovie3B);

            BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie3)
                .build();

            // Add all three movies to the list.
            List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
            myBatchStatementList.add(statementRequestMovie1);
            myBatchStatementList.add(statementRequestMovie2);
            myBatchStatementList.add(statementRequestMovie3);
```

```
            BatchExecuteStatementRequest batchRequest =
 BatchExecuteStatementRequest.builder()
                .statements(myBatchStatementList)
                .build();

            BatchExecuteStatementResponse response =
 ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: "+ response.toString());
            System.out.println("Added new movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItemBatch(DynamoDbClient ddb){
        String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Update three records.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();
```

```
        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: "+ response.toString());
            System.out.println("Updated three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb){
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
```

```
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName +" was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient ddb,
String statement, List<AttributeValue> parameters ) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
```

```
    }
}
```

- For API details, see BatchExecuteStatement in *AWS SDK for Java 2.x API Reference*.

Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class ScenarioPartiQ {

    public static void main(String [] args) throws IOException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <fileName>\n\n" +
            "Where:\n" +
            "    fileName - The path to the moviedata.json file that you can download
 from the Amazon DynamoDB Developer Guide.\n" ;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }

       String fileName = args[0];
       String tableName = "MoviesPartiQ";
       ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
       Region region = Region.US_EAST_1;
       DynamoDbClient ddb = DynamoDbClient.builder()
           .credentialsProvider(credentialsProvider)
           .region(region)
           .build();

       System.out.println("******* Creating an Amazon DynamoDB table named MoviesPartiQ
 with a key named year and a sort key named title.");
       createTable(ddb, tableName);

       System.out.println("******* Loading data into the MoviesPartiQ table.");
       loadData(ddb, fileName);

       System.out.println("******* Getting data from the MoviesPartiQ table.");
       getItem(ddb);

       System.out.println("******* Putting a record into the MoviesPartiQ table.");
       putRecord(ddb);
```

```
        System.out.println("******* Updating a record.");
        updateTableItem(ddb);

        System.out.println("******* Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("******* Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " +newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

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

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String fileName) throws IOException
{

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
'info' : ?}";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0 ;
        List<AttributeValue> parameters = new ArrayList<>();
        while (iter.hasNext()) {

            // Add 200 movies to the table.
            if (t == 200)
                break ;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf(year))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s(title)
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s(info)
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            // Insert the movie into the Amazon DynamoDB table.
            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added Movie " +title);

            parameters.remove(att1);
            parameters.remove(att2);
            parameters.remove(att3);
            t++;
        }
    }

    public static void getItem(DynamoDbClient ddb) {

        String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n("2012")
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();
```

```
        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: "+ response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added new movie.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItem(DynamoDbClient ddb){

        String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
 year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
 sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: "+ response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName +" was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient ddb,
 String statement, List<AttributeValue> parameters ) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse executeStatementResult)
 {
        System.out.println("ExecuteStatement successful: "+
 executeStatementResult.toString());
    }
}
```

- For API details, see ExecuteStatement in *AWS SDK for Java 2.x API Reference*.

Amazon EC2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon EC2.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon EC2

The following code examples show how to get started using Amazon EC2.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
            for(SecurityGroup group : response.securityGroups()) {
               System.out.printf(
                   "Found Security Group with id %s, " +
                           "vpc id %s " +
                           "and description %s",
                   group.groupId(),
                   group.vpcId(),
                   group.description());
            }

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
   }
```

- For API details, see DescribeSecurityGroups in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Allocate an Elastic IP address

The following code example shows how to allocate an Elastic IP address for Amazon EC2.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String getAllocateAddress( Ec2Client ec2, String instanceId) {

       try {
           AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
               .domain(DomainType.VPC)
               .build();

           AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
           String allocationId = allocateResponse.allocationId();
           AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
              .instanceId(instanceId)
              .allocationId(allocationId)
              .build();

           AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
           return associateResponse.associationId();

       } catch (Ec2Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
       }
       return "";
   }
```

- For API details, see AllocateAddress in *AWS SDK for Java 2.x API Reference*.

Associate an Elastic IP address with an instance

The following code example shows how to associate an Elastic IP address with an Amazon EC2 instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
       try {
           AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
               .instanceId(instanceId)
               .allocationId(allocationId)
               .build();
```

```
            AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
            return associateResponse.associationId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see AssociateAddress in *AWS SDK for Java 2.x API Reference*.

Create a security group

The following code example shows how to create an Amazon EC2 security group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createEC2SecurityGroup( Ec2Client ec2,String groupName, String
groupDesc, String vpcId) {
        try {

            CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                .groupName(groupName)
                .description(groupDesc)
                .vpcId(vpcId)
                .build();

            CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);

            IpRange ipRange = IpRange.builder()
                .cidrIp("0.0.0.0/0").build();

            IpPermission ipPerm = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(80)
                .fromPort(80)
                .ipRanges(ipRange)
                .build();

            IpPermission ipPerm2 = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(22)
                .fromPort(22)
                .ipRanges(ipRange)
                .build();

            AuthorizeSecurityGroupIngressRequest authRequest =
                AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

            AuthorizeSecurityGroupIngressResponse authResponse =
```

```
            ec2.authorizeSecurityGroupIngress(authRequest);
            System.out.printf("Successfully added ingress policy to Security Group %s",
groupName);
            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateSecurityGroup in *AWS SDK for Java 2.x API Reference*.

Create a security key pair

The following code example shows how to create a security key pair for Amazon EC2.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createEC2KeyPair(Ec2Client ec2,String keyName ) {
        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                .keyName(keyName)
                .build();

            ec2.createKeyPair(request);
            System.out.printf("Successfully created key pair named %s", keyName);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
     }
```

- For API details, see CreateKeyPair in *AWS SDK for Java 2.x API Reference*.

Create and run an instance

The following code example shows how to create and run an Amazon EC2 instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createEC2Instance(Ec2Client ec2,String name, String amiId ) {

        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .imageId(amiId)
            .instanceType(InstanceType.T1_MICRO)
```

```
            .maxCount(1)
            .minCount(1)
            .build();

        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceId = response.instances().get(0).instanceId();
        Tag tag = Tag.builder()
            .key("Name")
            .value(name)
            .build();

        CreateTagsRequest tagRequest = CreateTagsRequest.builder()
            .resources(instanceId)
            .tags(tag)
            .build();

        try {
            ec2.createTags(tagRequest);
            System.out.printf(  "Successfully started EC2 Instance %s based on AMI %s",
instanceId, amiId);
            return instanceId;

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- For API details, see RunInstances in *AWS SDK for Java 2.x API Reference*.

Delete a security group

The following code example shows how to delete an Amazon EC2 security group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {

        try {
            DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
                .groupId(groupId)
                .build();

            ec2.deleteSecurityGroup(request);
            System.out.printf("Successfully deleted Security Group with id %s",
groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
     }
```

- For API details, see DeleteSecurityGroup in *AWS SDK for Java 2.x API Reference*.

Delete a security key pair

The following code example shows how to delete an Amazon EC2 security key pair.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteKeys(Ec2Client ec2, String keyPair) {

        try {
            DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
                .keyName(keyPair)
                .build();

            ec2.deleteKeyPair(request);
            System.out.printf("Successfully deleted key pair named %s", keyPair);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteKeyPair in *AWS SDK for Java 2.x API Reference*.

Describe instances

The following code example shows how to describe Amazon EC2 instances.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String describeEC2Instances( Ec2Client ec2, String newInstanceId) {
        try {
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .instanceIds(newInstanceId)
                .build();

            while (!isRunning) {
                DescribeInstancesResponse response = ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") ==0) {
                     System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                     System.out.println("Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                     System.out.println("Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                     pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
```

```
                    System.out.println("Instance address is " + pubAddress);
                    isRunning = true;
                }
            }
            return pubAddress;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see DescribeInstances in *AWS SDK for Java 2.x API Reference*.

Disassociate an Elastic IP address from an instance

The following code example shows how to disassociate an Elastic IP address from an Amazon EC2 instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void disassociateAddress(Ec2Client ec2, String associationId) {
        try {
            DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
                .associationId(associationId)
                .build();

            ec2.disassociateAddress(addressRequest);
            System.out.println("You successfully disassociated the address!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DisassociateAddress in *AWS SDK for Java 2.x API Reference*.

Get data about a security group

The following code example shows how to get data about an Amazon EC2 security group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
```

```
                .groupIds(groupId)
                .build();

            DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
            for(SecurityGroup group : response.securityGroups()) {
                System.out.println( "Found Security Group with Id " +group.groupId() +"
and group VPC "+ group.vpcId());
            }

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DescribeSecurityGroups in *AWS SDK for Java 2.x API Reference*.

Get data about instance types

The following code example shows how to get data about Amazon EC2 instance types.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Get a list of instance types.
    public static String getInstanceTypes(Ec2Client ec2) {
        String instanceType="";
        try {
            List<Filter> filters = new ArrayList<>();
            Filter filter = Filter.builder()
                .name("processor-info.supported-architecture")
                .values("arm64")
                .build();

            filters.add(filter);
            DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
                .filters(filters)
                .maxResults(10)
                .build();

            DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
            List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
            for (InstanceTypeInfo type: instanceTypes) {
                System.out.println("The memory information of this type is
"+type.memoryInfo().sizeInMiB());
                System.out.println("Network information is
"+type.networkInfo().toString());
                instanceType = type.instanceType().toString();
            }

            return instanceType;

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }
```

- For API details, see DescribeInstanceTypes in *AWS SDK for Java 2.x API Reference*.

List security key pairs

The following code example shows how to list Amazon EC2 security key pairs.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeEC2Keys( Ec2Client ec2){

        try {
            DescribeKeyPairsResponse response = ec2.describeKeyPairs();
            response.keyPairs().forEach(keyPair -> System.out.printf(
                "Found key pair with name %s " +
                   "and fingerprint %s",
                    keyPair.keyName(),
                    keyPair.keyFingerprint())
            );

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
         }
    }
```

- For API details, see DescribeKeyPairs in *AWS SDK for Java 2.x API Reference*.

Release an Elastic IP address

The following code example shows how to release an Elastic IP address.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void releaseEC2Address(Ec2Client ec2,String allocId) {

        try {
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
                .allocationId(allocId)
                .build();

            ec2.releaseAddress(request);
            System.out.printf("Successfully released elastic IP address %s", allocId);

        } catch (Ec2Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ReleaseAddress in *AWS SDK for Java 2.x API Reference*.

Set inbound rules for a security group

The following code example shows how to set inbound rules for an Amazon EC2 security group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createSecurityGroup(Ec2Client ec2,String groupName, String
groupDesc, String vpcId, String myIpAddress) {
       try {
           CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
               .groupName(groupName)
               .description(groupDesc)
               .vpcId(vpcId)
               .build();

           CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
           IpRange ipRange = IpRange.builder()
               .cidrIp(myIpAddress+"/0")
               .build();

           IpPermission ipPerm = IpPermission.builder()
               .ipProtocol("tcp")
               .toPort(80)
               .fromPort(80)
               .ipRanges(ipRange)
               .build();

           IpPermission ipPerm2 = IpPermission.builder()
               .ipProtocol("tcp")
               .toPort(22)
               .fromPort(22)
               .ipRanges(ipRange)
               .build();

           AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
               .groupName(groupName)
               .ipPermissions(ipPerm, ipPerm2)
               .build();

           ec2.authorizeSecurityGroupIngress(authRequest);
           System.out.println("Successfully added ingress policy to security group
"+groupName);
           return resp.groupId();

       } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
        return "";
    }
```

- For API details, see AuthorizeSecurityGroupIngress in *AWS SDK for Java 2.x API Reference*.

Start an instance

The following code example shows how to start an Amazon EC2 instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void startInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        StartInstancesRequest request = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to run. This will
take a few minutes.");
        ec2.startInstances(request);
        DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance "+instanceId);
    }
```

- For API details, see StartInstances in *AWS SDK for Java 2.x API Reference*.

Stop an instance

The following code example shows how to stop an Amazon EC2 instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void stopInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();
        StopInstancesRequest request = StopInstancesRequest.builder()
```

```
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

       WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
       waiterResponse.matched().response().ifPresent(System.out::println);
       System.out.println("Successfully stopped instance "+instanceId);
    }
```

- For API details, see StopInstances in *AWS SDK for Java 2.x API Reference*.

Terminate an instance

The following code example shows how to terminate an Amazon EC2 instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void terminateEC2( Ec2Client ec2, String instanceID) {

        try{
            TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
                .instanceIds(instanceID)
                .build();

            TerminateInstancesResponse response = ec2.terminateInstances(ti);
            List<InstanceStateChange> list = response.terminatingInstances();
            for (InstanceStateChange sc : list) {
                System.out.println("The ID of the terminated instance is " +
sc.instanceId());
            }

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see TerminateInstances in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with instances

The following code example shows how to:

- Create a key pair and security group.

- Select an Amazon Machine Image (AMI) and compatible instance type, then create an instance.
- Stop and restart the instance.
- Associate an Elastic IP address with your instance.
- Connect to your instance with SSH, then clean up resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
 architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "   <keyName> <fileName> <groupName> <groupDesc> <vpcId>\n\n" +
            "Where:\n" +
            "   keyName -  A key pair name (for example, TestKeyPair). \n\n" +
            "   fileName -  A file name where the key information is written to. \n\n"
 +
            "   groupName - The name of the security group. \n\n" +
            "   groupDesc - The description of the security group. \n\n" +
            "   vpcId - A VPC Id value. You can get this value from the AWS Management
 Console. \n\n" +
            "   myIpAddress - The IP address of your development machine. \n\n" ;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String keyName = args[0];
        String fileName = args[1];
        String groupName = args[2];
        String groupDesc = args[3];
        String vpcId = args[4];
        String myIpAddress = args[5];

        Region region = Region.US_WEST_2;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an RSA key pair and save the private key material
as a .pem file.");
        createKeyPair(ec2, keyName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List key pairs.");
        describeKeys(ec2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a security group.");
        String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Display security group info for the newly created
security group.");
        describeSecurityGroups(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one with
amzn2 in the name.");
        String instanceId = getParaValues(ssmClient);
        System.out.println("The instance Id is "+instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Get more information about an amzn2 image.");
        String amiValue = describeImage(ec2, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of instance types.");
        String instanceType = getInstanceTypes(ec2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an instance.");
```

```
        String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue );
        System.out.println("The instance Id is "+newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Display information about the running instance. ");
        String ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Stop the instance and use a waiter.");
        stopInstance(ec2, newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Start the instance and use a waiter.");
        startInstance(ec2, newInstanceId);
        ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
        String allocationId = allocateAddress(ec2);
        System.out.println("The allocation Id value is "+allocationId);
        String associationId = associateAddress(ec2, newInstanceId, allocationId);
        System.out.println("The associate Id value is "+associationId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Describe the instance again.");
        ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Disassociate and release the Elastic IP address.");
        disassociateAddress(ec2, associationId);
        releaseEC2Address(ec2, allocationId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Terminate the instance and use a waiter.");
        terminateEC2(ec2, newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the security group.");
        deleteEC2SecGroup(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Delete the key.");
        deleteKeys(ec2, keyName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("You successfully completed the Amazon EC2 scenario.");
        System.out.println(DASHES);
        ec2.close();
```

```
    }

    public static void deleteEC2SecGroup(Ec2Client ec2,String groupId) {
        try {
            DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
                .groupId(groupId)
                .build();

            ec2.deleteSecurityGroup(request);
            System.out.println("Successfully deleted security group with Id "
+groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateEC2(Ec2Client ec2, String instanceId) {
        try{
            Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();

            TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

            System.out.println("Use an Ec2Waiter to wait for the instance to terminate.
This will take a few minutes.");
            ec2.terminateInstances(ti);
            DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

            WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceTerminated(instanceRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Successfully started instance "+instanceId);
            System.out.println(instanceId +" is terminated!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKeys(Ec2Client ec2, String keyPair) {
        try {
            DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
                .keyName(keyPair)
                .build();

            ec2.deleteKeyPair(request);
            System.out.println("Successfully deleted key pair named "+keyPair);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void releaseEC2Address(Ec2Client ec2,String allocId) {
        try {
```

```
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
                .allocationId(allocId)
                .build();

            ec2.releaseAddress(request);
            System.out.println("Successfully released Elastic IP address "+allocId);
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disassociateAddress(Ec2Client ec2, String associationId) {
        try {
            DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
                .associationId(associationId)
                .build();

            ec2.disassociateAddress(addressRequest);
            System.out.println("You successfully disassociated the address!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
        try {
            AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
                .instanceId(instanceId)
                .allocationId(allocationId)
                .build();

            AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
            return associateResponse.associationId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String allocateAddress(Ec2Client ec2) {
        try {
            AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
                .domain(DomainType.VPC)
                .build();

            AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
            return allocateResponse.allocationId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void startInstance(Ec2Client ec2, String instanceId) {
```

```
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        StartInstancesRequest request = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to run. This will
take a few minutes.");
        ec2.startInstances(request);
        DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance "+instanceId);
    }

    public static void stopInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();
        StopInstancesRequest request = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

       WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
       waiterResponse.matched().response().ifPresent(System.out::println);
       System.out.println("Successfully stopped instance "+instanceId);
    }

    public static String describeEC2Instances( Ec2Client ec2, String newInstanceId) {
        try {
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .instanceIds(newInstanceId)
                .build();

            while (!isRunning) {
                DescribeInstancesResponse response = ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") ==0) {
                     System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                     System.out.println("Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                     System.out.println("Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                     pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
                     System.out.println("Instance address is " + pubAddress);
```

```
                    isRunning = true;
                }
            }
            return pubAddress;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName, String amiId ) {
        try {
            RunInstancesRequest runRequest = RunInstancesRequest.builder()
                .instanceType(instanceType)
                .keyName(keyName)
                .securityGroups(groupName)
                .maxCount(1)
                .minCount(1)
                .imageId(amiId)
                .build();

            RunInstancesResponse response = ec2.runInstances(runRequest);
            String instanceId = response.instances().get(0).instanceId();
            System.out.println("Successfully started EC2 instance "+instanceId +" based
on AMI "+amiId);
            return instanceId;

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

    // Get a list of instance types.
    public static String getInstanceTypes(Ec2Client ec2) {
        String instanceType="";
        try {
            List<Filter> filters = new ArrayList<>();
            Filter filter = Filter.builder()
                .name("processor-info.supported-architecture")
                .values("arm64")
                .build();

            filters.add(filter);
            DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
                .filters(filters)
                .maxResults(10)
                .build();

            DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
            List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
            for (InstanceTypeInfo type: instanceTypes) {
                System.out.println("The memory information of this type is
"+type.memoryInfo().sizeInMiB());
                System.out.println("Network information is
"+type.networkInfo().toString());
                instanceType = type.instanceType().toString();
            }

            return instanceType;
```

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

    // Display the Description field that corresponds to the instance Id value.
    public static String describeImage(Ec2Client ec2, String instanceId) {
        try {
            DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
                .imageIds(instanceId)
                .build();

            DescribeImagesResponse response = ec2.describeImages(imagesRequest);
            System.out.println("The description of the first image is
"+response.images().get(0).description());
            System.out.println("The name of the first image is
"+response.images().get(0).name());

            // Return the image Id value.
            return response.images().get(0).imageId();

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

    // Get the Id value of an instance with amzn2 in the name.
    public static String getParaValues(SsmClient ssmClient) {
        try {
            GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
                .path("/aws/service/ami-amazon-linux-latest")
                .build();

            GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
            for (software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse
response : responses) {
                System.out.println("Test "+response.nextToken());
                List<Parameter> parameterList = response.parameters();
                for (Parameter para: parameterList) {
                    System.out.println("The name of the para is: "+para.name());
                    System.out.println("The type of the para is: "+para.type());
                    if (filterName(para.name())) {
                        return para.value();
                    }
                }
            }

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

    // Return true if the name has amzn2 in it. For example:
    // /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
    private static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
```

```
   }

   public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
       try {
           DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
               .groupIds(groupId)
               .build();

           DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
           for(SecurityGroup group : response.securityGroups()) {
               System.out.println( "Found Security Group with Id " +group.groupId() +"
and group VPC "+ group.vpcId());
           }

       } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static String createSecurityGroup(Ec2Client ec2,String groupName, String
groupDesc, String vpcId, String myIpAddress) {
       try {
           CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
               .groupName(groupName)
               .description(groupDesc)
               .vpcId(vpcId)
               .build();

           CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
           IpRange ipRange = IpRange.builder()
               .cidrIp(myIpAddress+"/0")
               .build();

           IpPermission ipPerm = IpPermission.builder()
               .ipProtocol("tcp")
               .toPort(80)
               .fromPort(80)
               .ipRanges(ipRange)
               .build();

           IpPermission ipPerm2 = IpPermission.builder()
               .ipProtocol("tcp")
               .toPort(22)
               .fromPort(22)
               .ipRanges(ipRange)
               .build();

           AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
               .groupName(groupName)
               .ipPermissions(ipPerm, ipPerm2)
               .build();

           ec2.authorizeSecurityGroupIngress(authRequest);
           System.out.println("Successfully added ingress policy to security group
"+groupName);
           return resp.groupId();

       } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
        return "";
    }

    public static void describeKeys( Ec2Client ec2){
        try {
            DescribeKeyPairsResponse response = ec2.describeKeyPairs();
            response.keyPairs().forEach(keyPair -> System.out.printf(
                "Found key pair with name %s " +
                    "and fingerprint %s",
                keyPair.keyName(),
                keyPair.keyFingerprint())
            );

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeyPair(Ec2Client ec2, String keyName, String fileName) {
        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                .keyName(keyName)
                .build();

            CreateKeyPairResponse response = ec2.createKeyPair(request);
            String content = response.keyMaterial();
            BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
            writer.write(content);
            writer.close();
            System.out.println("Successfully created key pair named "+keyName);

        } catch (Ec2Exception | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances

- TerminateInstances
- UnmonitorInstances

Amazon EC2 Auto Scaling examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon EC2 Auto Scaling.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a group

The following code example shows how to create an Auto Scaling group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                             String groupName,
                                             String launchTemplateName,
                                             String serviceLinkedRoleARN,
                                             String vpcZoneId) {

       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
               .launchTemplateName(launchTemplateName)
               .build();

           CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
               .autoScalingGroupName(groupName)
               .availabilityZones("us-east-1a")
               .launchTemplate(templateSpecification)
               .maxSize(1)
               .minSize(1)
               .vpcZoneIdentifier(vpcZoneId)
               .serviceLinkedRoleARN(serviceLinkedRoleARN)
               .build();

           autoScalingClient.createAutoScalingGroup(request);
```

```
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

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

- For API details, see CreateAutoScalingGroup in *AWS SDK for Java 2.x API Reference*.

Delete a group

The following code example shows how to delete an Auto Scaling group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build() ;

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;
            System.out.println("You successfully deleted "+groupName);

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

- For API details, see DeleteAutoScalingGroup in *AWS SDK for Java 2.x API Reference*.

Disable metrics collection for a group

The following code example shows how to disable CloudWatch metrics collection for an Auto Scaling group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
               .autoScalingGroupName(groupName)
               .metrics("GroupMaxSize")
               .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

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

- For API details, see DisableMetricsCollection in *AWS SDK for Java 2.x API Reference*.

Enable metrics collection for a group

The following code example shows how to enable CloudWatch metrics collection for an Auto Scaling group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
       try {

           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
               .autoScalingGroupName(groupName)
               .metrics("GroupMaxSize")
               .granularity("1Minute")
               .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

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

- For API details, see EnableMetricsCollection in *AWS SDK for Java 2.x API Reference*.

Get information about groups

The following code example shows how to get information about Auto Scaling groups.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String getAutoScaling( AutoScalingClient autoScalingClient, String
groupName) {
       try{
           String instanceId = "";
           DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
               .autoScalingGroupNames(groupName)
               .build();

           DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
           List<AutoScalingGroup> groups = response.autoScalingGroups();
           for (AutoScalingGroup group: groups) {
               System.out.println("The group name is " +
group.autoScalingGroupName());
               System.out.println("The group ARN is " + group.autoScalingGroupARN());

               List<Instance> instances = group.instances();
               for (Instance instance : instances) {
                   instanceId = instance.instanceId();
               }
           }
           return instanceId;
       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- For API details, see DescribeAutoScalingGroups in *AWS SDK for Java 2.x API Reference*.

Get information about instances

The following code example shows how to get information about Auto Scaling instances.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeAutoScalingInstance( AutoScalingClient
autoScalingClient, String id) {
       try {
           DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest =
DescribeAutoScalingInstancesRequest.builder()
               .instanceIds(id)
```

```
                .build();

            DescribeAutoScalingInstancesResponse response =
autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance:instances ) {
                System.out.println("The instance lifecycle state is:
"+instance.lifecycleState());
            }

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

- For API details, see DescribeAutoScalingInstances in *AWS SDK for Java 2.x API Reference*.

Get information about scaling activities

The following code example shows how to get information about Auto Scaling activities.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeScalingActivities(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response =
autoScalingClient.describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity: activities) {
                System.out.println("The activity Id is "+activity.activityId());
                System.out.println("The activity details are "+activity.details());
            }

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

- For API details, see DescribeScalingActivities in *AWS SDK for Java 2.x API Reference*.

Set the desired capacity of a group

The following code example shows how to set the desired capacity of an Auto Scaling group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void setDesiredCapacity(AutoScalingClient autoScalingClient, String
groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
               .autoScalingGroupName(groupName)
               .desiredCapacity(2)
               .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

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

- For API details, see SetDesiredCapacity in *AWS SDK for Java 2.x API Reference*.

Terminate an instance in a group

The following code example shows how to terminate an instance in an Auto Scaling group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId){
       try {
           TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
               .instanceId(instanceId)
               .shouldDecrementDesiredCapacity(false)
               .build();

           autoScalingClient.terminateInstanceInAutoScalingGroup(request);
           System.out.println("You have terminated instance "+instanceId);

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

- For API details, see TerminateInstanceInAutoScalingGroup in *AWS SDK for Java 2.x API Reference*.

Update a group

The following code example shows how to update the configuration for an Auto Scaling group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName, String serviceLinkedRoleARN) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
               .launchTemplateName(launchTemplateName)
               .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
               .maxSize(3)
               .serviceLinkedRoleARN(serviceLinkedRoleARN)
               .autoScalingGroupName(groupName)
               .launchTemplate(templateSpecification)
               .build();

           autoScalingClient.updateAutoScalingGroup(groupRequest);
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
               .autoScalingGroupNames(groupName)
               .build();

           WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupInService(groupsRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("You successfully updated the auto scaling group
"+groupName);

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

- For API details, see UpdateAutoScalingGroup in *AWS SDK for Java 2.x API Reference*.

Scenarios

Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group with a launch template and Availability Zones, and get information about running instances.
- Enable Amazon CloudWatch metrics collection.
- Update the group's desired capacity and wait for an instance to start.
- Terminate an instance in the group.

- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics, then clean up resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 *  Before running this SDK for Java (v2) code example, set up your development
 environment, including your credentials.
 *
 *  For more information, see the following documentation:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  In addition, create a launch template. For more information, see the following
 topic:
 *
 *  https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */


public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>\n
\n" +
            "Where:\n" +
            "    groupName - The name of the Auto Scaling group.\n" +
            "    launchTemplateName - The name of the launch template. \n" +
            "    serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
linked role that the Auto Scaling group uses.\n" +
            "    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
 instances in the Auto Scaling group can be created.\n" ;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String serviceLinkedRoleARN = args[2];
```

```
        String vpcZoneId = args[3];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Auto Scaling group named "+groupName);
        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
 serviceLinkedRoleARN, vpcZoneId);
        System.out.println("Wait 1 min for the resources, including the instance.
 Otherwise, an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient, groupName);
        if (instanceId.compareTo("") ==0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is "+instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value "+instanceId);
        describeAutoScalingInstance( autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection "+instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to 3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
 serviceLinkedRoleARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println("Wait 1 min for the resources, including the instance.
 Otherwise, an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
```

```
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for the
group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

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

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response =
autoScalingClient.describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity: activities) {
                System.out.println("The activity Id is "+activity.activityId());
                System.out.println("The activity details are "+activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient, String
groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                .autoScalingGroupName(groupName)
                .desiredCapacity(2)
                .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");
```

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

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                              String groupName,
                                              String launchTemplateName,
                                              String serviceLinkedRoleARN,
                                              String vpcZoneId) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(launchTemplateName)
                .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .availabilityZones("us-east-1a")
                .launchTemplate(templateSpecification)
                .maxSize(1)
                .minSize(1)
                .vpcZoneIdentifier(vpcZoneId)
                .serviceLinkedRoleARN(serviceLinkedRoleARN)
                .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance( AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest =
DescribeAutoScalingInstancesRequest.builder()
                .instanceIds(id)
                .build();

            DescribeAutoScalingInstancesResponse response =
autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance:instances ) {
                System.out.println("The instance lifecycle state is:
"+instance.lifecycleState());
            }

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

```
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .maxRecords(10)
                .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group: groups) {
                System.out.println("*** The service to use for the health checks: "+
group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try{
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group: groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " + group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is "
+instance.lifecycleState());
                }
            }

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

    public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {

            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
```

```
                .autoScalingGroupName(groupName)
                .metrics("GroupMaxSize")
                .granularity("1Minute")
                .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                .autoScalingGroupName(groupName)
                .metrics("GroupMaxSize")
                .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
        try {
            DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
            System.out.println("The max number of auto scaling groups is
"+response.maxNumberOfAutoScalingGroups());
            System.out.println("The current number of auto scaling groups is
"+response.numberOfAutoScalingGroups());

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName, String serviceLinkedRoleARN) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(launchTemplateName)
                .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                .maxSize(3)
                .serviceLinkedRoleARN(serviceLinkedRoleARN)
                .autoScalingGroupName(groupName)
                .launchTemplate(templateSpecification)
                .build();
```

```
            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter.waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
 "+groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
 autoScalingClient, String instanceId){
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
 TerminateInstanceInAutoScalingGroupRequest.builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance "+instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build() ;

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;
            System.out.println("You successfully deleted "+groupName);

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities

- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Amazon ECS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon ECS.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a cluster

The following code example shows how to create an Amazon ECS cluster.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createGivenCluster( EcsClient ecsClient, String clusterName) {

       try {
           ExecuteCommandConfiguration commandConfiguration =
ExecuteCommandConfiguration.builder()
               .logging(ExecuteCommandLogging.DEFAULT)
               .build();

           ClusterConfiguration clusterConfiguration = ClusterConfiguration.builder()
               .executeCommandConfiguration(commandConfiguration)
               .build();

           CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
               .clusterName(clusterName)
               .configuration(clusterConfiguration)
               .build();

           CreateClusterResponse response = ecsClient.createCluster(clusterRequest) ;
           return response.cluster().clusterArn();

       } catch (EcsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
```

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

- For API details, see CreateCluster in *AWS SDK for Java 2.x API Reference*.

Create a service

The following code example shows how to create an Amazon ECS service.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createNewService(EcsClient ecsClient,
                                          String clusterName,
                                          String serviceName,
                                          String securityGroups,
                                          String subnets,
                                          String taskDefinition) {

        try {
            AwsVpcConfiguration vpcConfiguration = AwsVpcConfiguration.builder()
                .securityGroups(securityGroups)
                .subnets(subnets)
                .build();

            NetworkConfiguration configuration = NetworkConfiguration.builder()
                .awsvpcConfiguration(vpcConfiguration)
                .build();

            CreateServiceRequest serviceRequest = CreateServiceRequest.builder()
                .cluster(clusterName)
                .networkConfiguration(configuration)
                .desiredCount(1)
                .launchType(LaunchType.FARGATE)
                .serviceName(serviceName)
                .taskDefinition(taskDefinition)
                .build();

            CreateServiceResponse response = ecsClient.createService(serviceRequest) ;
            return response.service().serviceArn();

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

- For API details, see CreateService in *AWS SDK for Java 2.x API Reference*.

Delete a service

The following code example shows how to delete an Amazon ECS service.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteSpecificService(EcsClient ecsClient, String clusterName,
String serviceArn) {

       try {
           DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
               .cluster(clusterName)
               .service(serviceArn)
               .build();

           ecsClient.deleteService(serviceRequest);
           System.out.println("The Service was successfully deleted");

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

- For API details, see DeleteService in *AWS SDK for Java 2.x API Reference*.

Describe clusters

The following code example shows how to describe your Amazon ECS clusters.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void descCluster(EcsClient ecsClient, String clusterArn) {

       try {
           DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
               .clusters(clusterArn)
               .build();

           DescribeClustersResponse response =
ecsClient.describeClusters(clustersRequest);
           List<Cluster> clusters = response.clusters();
           for (Cluster cluster: clusters) {
               System.out.println("The cluster name is "+cluster.clusterName());
           }

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

- For API details, see DescribeClusters in *AWS SDK for Java 2.x API Reference*.

Describe tasks

The following code example shows how to describe your Amazon ECS tasks.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
taskId) {

       try {
           DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
              .cluster(clusterArn)
              .tasks(taskId)
              .build();

           DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
           List<Task> tasks = response.tasks();
           for (Task task: tasks) {
               System.out.println("The task ARN is "+task.taskDefinitionArn());
           }

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

- For API details, see DescribeTasks in *AWS SDK for Java 2.x API Reference*.

List clusters

The following code example shows how to list your Amazon ECS clusters.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listAllClusters(EcsClient ecsClient) {

       try {
           ListClustersResponse response = ecsClient.listClusters();
           List<String> clusters = response.clusterArns();
           for (String cluster: clusters) {
               System.out.println("The cluster arn is "+cluster);
           }

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

- For API details, see ListClusters in *AWS SDK for Java 2.x API Reference*.

Update a service

The following code example shows how to update an Amazon ECS service.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void updateSpecificService( EcsClient ecsClient, String clusterName,
String serviceArn) {

       try {
           UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
               .cluster(clusterName)
               .service(serviceArn)
               .desiredCount(0)
               .build();

           ecsClient.updateService(serviceRequest);
           System.out.println("The service was modified");

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

- For API details, see UpdateService in *AWS SDK for Java 2.x API Reference*.

EventBridge examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with EventBridge.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a rule

The following code example shows how to create an Amazon EventBridge rule.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a scheduled rule.

```
       public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {

           try {
               PutRuleRequest ruleRequest = PutRuleRequest.builder()
                   .name(ruleName)
                   .eventBusName("default")
                   .scheduleExpression(cronExpression)
                   .state("ENABLED")
                   .description("A test rule that runs on a schedule created by the
Java API")
                   .build();

               PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
               System.out.println("The ARN of the new rule is "+
ruleResponse.ruleArn());

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

- For API details, see PutRule in *AWS SDK for Java 2.x API Reference*.

Delete a rule

The following code example shows how to delete an Amazon EventBridge rule.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteEBRule(EventBridgeClient eventBrClient, String ruleName) {

       try {
           DisableRuleRequest disableRuleRequest = DisableRuleRequest.builder()
               .name(ruleName)
               .eventBusName("default")
               .build();

           eventBrClient.disableRule(disableRuleRequest);
           DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .eventBusName("default")
               .build();

           eventBrClient.deleteRule(ruleRequest);
           System.out.println("Rule "+ruleName + " was successfully deleted!");

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

- For API details, see DeleteRule in *AWS SDK for Java 2.x API Reference*.

Send events

The following code example shows how to send Amazon EventBridge events.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void putEBEvents(EventBridgeClient eventBrClient, String resourceArn,
String resourceArn2 ) {

       try {
           // Populate a List with the resource ARN values.
           List<String> resources = new ArrayList<>();
           resources.add(resourceArn);
           resources.add(resourceArn2);

           PutEventsRequestEntry reqEntry = PutEventsRequestEntry.builder()
               .resources(resources)
               .source("com.mycompany.myapp")
               .detailType("myDetailType")
               .detail("{ \"key1\": \"value1\", \"key2\": \"value2\" }")
               .build();

           PutEventsRequest eventsRequest = PutEventsRequest.builder()
               .entries(reqEntry)
               .build();

           PutEventsResponse result = eventBrClient.putEvents(eventsRequest);
           for (PutEventsResultEntry resultEntry : result.entries()) {
               if (resultEntry.eventId() != null) {
                   System.out.println("Event Id: " + resultEntry.eventId());
               } else {
                   System.out.println("Injection failed with Error Code: " +
resultEntry.errorCode());
               }
           }

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

- For API details, see PutEvents in *AWS SDK for Java 2.x API Reference*.

AWS Glue examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Glue.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a crawler

The following code example shows how to create an AWS Glue crawler.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void createGlueCrawler(GlueClient glueClient,
                                     String iam,
                                     String s3Path,
                                     String cron,
                                     String dbName,
                                     String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.
        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);

        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName +" was successfully created");

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

- For API details, see CreateCrawler in *AWS SDK for Java 2.x API Reference*.

Get a crawler

The following code example shows how to get an AWS Glue crawler.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName) {

      try {
          GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
              .name(crawlerName)
              .build();

            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            Instant createDate = response.crawler().creationTime();

            // Convert the Instant to readable date
            DateTimeFormatter formatter =
                  DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                          .withLocale( Locale.US)
                          .withZone( ZoneId.systemDefault() );

            formatter.format( createDate );
            System.out.println("The create date of the Crawler is " + createDate );

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

- For API details, see GetCrawler in *AWS SDK for Java 2.x API Reference*.

Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getSpecificDatabase(GlueClient glueClient, String databaseName)
{

        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                .name(databaseName)
                .build();

            GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
```

```
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " + createDate);

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

- For API details, see GetDatabase in *AWS SDK for Java 2.x API Reference*.

Get tables from a database

The following code example shows how to get tables from a database in the AWS Glue Data Catalog.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName ) {

        try {
            GetTableRequest tableRequest = GetTableRequest.builder()
                .databaseName(dbName)
                .name(tableName)
                .build();

            GetTableResponse tableResponse = glueClient.getTable(tableRequest);
            Instant createDate = tableResponse.table().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                .withLocale( Locale.US)
                .withZone( ZoneId.systemDefault() );

            formatter.format( createDate );
            System.out.println("The create date of the table is " + createDate );

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

- For API details, see GetTables in *AWS SDK for Java 2.x API Reference*.

Start a crawler

The following code example shows how to start an AWS Glue crawler.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void startSpecificCrawler(GlueClient glueClient, String crawlerName)
 {

        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);

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

- For API details, see StartCrawler in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with crawlers and jobs

The following code example shows how to:

- Create a crawler that crawls a public Amazon S3 bucket and generates a database of CSV-formatted metadata.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create a job to extract CSV data from the S3 bucket, transform the data, and load JSON-formatted output into another S3 bucket.
- List information about job runs, view transformed data, and clean up resources.

For more information, see Tutorial: Getting started with AWS Glue Studio.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 *
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
            "Usage:\n" +
            "    <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> \n\n" +
            "Where:\n" +
            "    iam - The ARN of the IAM role that has AWS Glue and S3 permissions.
 \n" +
            "    s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).\n" +
            "    cron - A cron expression used to specify the schedule  (i.e., cron(15
 12 * * ? *).\n" +
            "    dbName - The database name. \n" +
            "    crawlerName - The name of the crawler. \n" +
            "    jobName - The name you assign to this job definition."+
            "    scriptLocation - The Amazon S3 path to a script that runs a job." +
            "    locationUri - The location of the database" +
            "    bucketNameSc - The Amazon S3 bucket name used when creating a job" ;

       if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
       }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
```

```
        createDatabase(glueClient, dbName, locationUri);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        getSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        startSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        getSpecificDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName = getGlueTables(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        createJob(glueClient, jobName, iam, scriptLocation);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc );
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the "+crawlerName +" to stop");
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
 locationUri ) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
                .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                .databaseInput(input)
                .build();

            glueClient.createDatabase(request);
            System.out.println(dbName +" was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
                                         String cron,
                                         String dbName,
                                         String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName +" was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName) {
        try {
```

```
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            boolean ready = false;
            while(!ready) {
                GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true ;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

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

    public static void startSpecificCrawler(GlueClient glueClient, String crawlerName)
{
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName +" was successfully started!");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificDatabase(GlueClient glueClient, String databaseName)
{
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                .name(databaseName)
                .build();

            GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                .withLocale( Locale.US)
                .withZone( ZoneId.systemDefault() );

            formatter.format( createDate );
            System.out.println("The create date of the database is " + createDate );

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getGlueTables(GlueClient glueClient, String dbName){
        String myTableName = "";
        try {
```

```
            GetTablesRequest tableRequest = GetTablesRequest.builder()
                .databaseName(dbName)
                .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return myTableName;
    }

    public static void startJob(GlueClient glueClient, String jobName, String
 inputDatabase, String inputTable, String outBucket) {
        try {
            Map<String,String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is "+
 response.responseMetadata().requestId());

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createJob(GlueClient glueClient, String jobName, String iam,
 String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();
```

```
            glueClient.createJob(jobRequest);
            System.out.println(jobName +" was successfully created.");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job: jobs) {
                System.out.println("Job name is : "+job.name());
                System.out.println("The job worker type is :
"+job.workerType().name());
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false ;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
```

```
                }
            }

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

    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName +" was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName +" was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSpecificCrawler(GlueClient glueClient, String crawlerName)
 {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest = DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName +" was deleted");

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob

- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

IAM examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with IAM.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

       try {
           ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
               .roleName(roleName)
               .build();

           ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
           List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
```

```
            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy: attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn)==0) {
                    System.out.println(roleName + " policy is already attached to this
role.");
                    return;
                }
            }

           AttachRolePolicyRequest attachRequest = AttachRolePolicyRequest.builder()
               .roleName(roleName)
               .policyArn(policyArn)
               .build();

           iam.attachRolePolicy(attachRequest);

           System.out.println("Successfully attached policy " + policyArn +
                " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

- For API details, see AttachRolePolicy in *AWS SDK for Java 2.x API Reference*.

Create a policy

The following code example shows how to create an IAM policy.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createIAMPolicy(IamClient iam, String policyName ) {

        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                .policyName(policyName)
                .policyDocument(PolicyDocument)
                .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                .policyArn(response.policy().arn())
                .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
            waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
```

```
            return response.policy().arn();

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

- For API details, see CreatePolicy in *AWS SDK for Java 2.x API Reference*.

Create a role

The following code example shows how to create an IAM role.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createIAMRole(IamClient iam, String rolename, String
 fileLocation ) throws Exception {

        try {
            JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(jsonObject.toJSONString())
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is "+response.role().arn());

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

    public static Object readJsonSimpleDemo(String filename) throws Exception {
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
```

- For API details, see CreateRole in *AWS SDK for Java 2.x API Reference*.

Create a user

The following code example shows how to create an IAM user.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createIAMUser(IamClient iam, String username ) {

        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                .userName(username)
                .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created
            GetUserRequest userRequest = GetUserRequest.builder()
                .userName(response.user().userName())
                .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

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

- For API details, see CreateUser in *AWS SDK for Java 2.x API Reference*.

Create an access key

The following code example shows how to create an IAM access key.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createIAMAccessKey(IamClient iam,String user) {

        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
```

```
            return response.accessKey().accessKeyId();

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

- For API details, see CreateAccessKey in *AWS SDK for Java 2.x API Reference*.

Create an alias for an account

The following code example shows how to create an alias for an IAM account.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createIAMAccountAlias(IamClient iam, String alias) {

        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
                .accountAlias(alias)
                .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

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

- For API details, see CreateAccountAlias in *AWS SDK for Java 2.x API Reference*.

Delete a policy

The following code example shows how to delete an IAM policy.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteIAMPolicy(IamClient iam,String policyARN) {

        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(policyARN)
                .build();

            iam.deletePolicy(request);
```

```
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

- For API details, see DeletePolicy in *AWS SDK for Java 2.x API Reference*.

Delete a user

The following code example shows how to delete an IAM user.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteIAMUser(IamClient iam, String userName) {

        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                .userName(userName)
                .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

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

- For API details, see DeleteUser in *AWS SDK for Java 2.x API Reference*.

Delete an access key

The following code example shows how to delete an IAM access key.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {

    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);

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

- For API details, see DeleteAccessKey in *AWS SDK for Java 2.x API Reference*.

Delete an account alias

The following code example shows how to delete an IAM account alias.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void deleteIAMAccountAlias(IamClient iam, String alias ) {

    try {
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.deleteAccountAlias(request);
        System.out.println("Successfully deleted account alias " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see DeleteAccountAlias in *AWS SDK for Java 2.x API Reference*.

Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void detachPolicy(IamClient iam, String roleName, String policyArn )
{

        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                " from role " + roleName);

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

- For API details, see DetachRolePolicy in *AWS SDK for Java 2.x API Reference*.

List a user's access keys

The following code example shows how to list a user's IAM access keys.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listKeys( IamClient iam,String userName ){

        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if(newMarker == null) {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                        .userName(userName)
                        .marker(newMarker)
                        .build();

                    response = iam.listAccessKeys(request);
```

```
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

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

- For API details, see ListAccessKeys in *AWS SDK for Java 2.x API Reference*.

List account aliases

The following code example shows how to list IAM account aliases.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAliases(IamClient iam) {

        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }

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

- For API details, see ListAccountAliases in *AWS SDK for Java 2.x API Reference*.

List users

The following code example shows how to list IAM users.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listAllUsers(IamClient iam ) {

       try {

           boolean done = false;
           String newMarker = null;

           while(!done) {
               ListUsersResponse response;
               if (newMarker == null) {
                   ListUsersRequest request = ListUsersRequest.builder().build();
                   response = iam.listUsers(request);
               } else {
                   ListUsersRequest request = ListUsersRequest.builder()
                       .marker(newMarker)
                       .build();

                   response = iam.listUsers(request);
               }

               for(User user : response.users()) {
                   System.out.format("\n Retrieved user %s", user.userName());
                   AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
                   if (permissionsBoundary != null)
                       System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
               }

               if(!response.isTruncated()) {
                   done = true;
               } else {
                   newMarker = response.marker();
               }
           }

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

- For API details, see ListUsers in *AWS SDK for Java 2.x API Reference*.

Update a user

The following code example shows how to update an IAM user.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void updateIAMUser(IamClient iam, String curName,String newName ) {

        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                .userName(curName)
                .newUserName(newName)
                .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s", newName);

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

- For API details, see UpdateUser in *AWS SDK for Java 2.x API Reference*.

Update an access key

The following code example shows how to update an IAM access key.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                 statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                .accessKeyId(accessId)
                .userName(username)
                .status(statusType)
                .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s to" +
```

```
                        "status %s for user %s", accessId, status, username);

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

- For API details, see UpdateAccessKey in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a user and assume a role

The following code example shows how to create a user and assume a role.

> **Warning**
> To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as AWS IAM Identity Center (successor to AWS Single Sign-On).

- Create a user with no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list S3 buckets using temporary credentials, then clean up resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create functions that wrap IAM user actions.

```
/*
  To run this Java V2 code example, set up your development environment, including your
 credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3 Service
 client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
```

```
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;
    public static void main(String[] args) throws Exception {

        final String usage = "\n" +
            "Usage:\n" +
            "    <username> <policyName> <roleName> <roleSessionName> <bucketName> \n
\n" +
            "Where:\n" +
            "    username - The name of the IAM user to create. \n\n" +
            "    policyName - The name of the policy to create. \n\n" +
            "    roleName - The name of the role to create. \n\n" +
            "    roleSessionName - The name of the session required for the assumeRole
 operation. \n\n" +
            "    bucketName - The name of the Amazon S3 bucket from which objects are
 read. \n\n";

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            " \"AWS\": \"" + userArn + "\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}";
```

```
        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the role.");
        System.out.println("Perform an Amazon S3 Service operation using the temporary
credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey );
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                .userName(user)
                .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                .userName(username)
                .build();
```

```
            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                .userName(response.user().userName())
                .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(json)
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());
            return response.role().arn();

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

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                .policyName(policyName)
                .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                .policyArn(response.policy().arn())
                .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
            waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

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

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
```

```
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                .roleName(roleName)
                .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to this
role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest = AttachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(policyArn)
                .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to role
" + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal, String keySecret) {

        // Use the creds of the new IAM user that was created in this code example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(StaticCredentialsProvider.create(credentials))
            .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds retrieved
by invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()

.credentialsProvider(StaticCredentialsProvider.create(AwsSessionCredentials.create(key,
secKey, secToken)))
                .region(region)
                .build();
```

```
            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
    public static void deleteRole(IamClient iam, String roleName, String polArn) {

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(polArn)
                .roleName(roleName)
                .build();

            iam.detachRolePolicy(rolePolicyRequest);

            // Delete the policy.
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(polArn)
                .build();

            iam.deletePolicy(request);
            System.out.println("*** Successfully deleted " + polArn);

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKey(IamClient iam ,String username, String accessKey ) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                .accessKeyId(accessKey)
                .userName(username)
                .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                .userName(userName)
                .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

AWS KMS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS KMS.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a grant for a key

The following code example shows how to create a grant for a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createGrant(KmsClient kmsClient, String keyId, String
granteePrincipal, String operation) {

       try {
           CreateGrantRequest grantRequest = CreateGrantRequest.builder()
               .keyId(keyId)
               .granteePrincipal(granteePrincipal)
               .operationsWithStrings(operation)
               .build();

           CreateGrantResponse response = kmsClient.createGrant(grantRequest);
           return response.grantId();

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

- For API details, see CreateGrant in *AWS SDK for Java 2.x API Reference*.

Create a key

The following code example shows how to create an AWS KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createKey(KmsClient kmsClient, String keyDesc) {

       try {
           CreateKeyRequest keyRequest = CreateKeyRequest.builder()
               .description(keyDesc)
               .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
               .keyUsage("ENCRYPT_DECRYPT")
               .build();

           CreateKeyResponse result = kmsClient.createKey(keyRequest);
           System.out.printf("Created a customer key with id \"%s\"%n",
result.keyMetadata().arn());
           return result.keyMetadata().keyId();

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

- For API details, see CreateKey in *AWS SDK for Java 2.x API Reference*.

Create an alias for a key

The following code example shows how to create an alias for a KMS key key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {

       try {
           CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
               .aliasName(aliasName)
               .targetKeyId(targetKeyId)
               .build();

           kmsClient.createAlias(aliasRequest);

       } catch (KmsException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see CreateAlias in *AWS SDK for Java 2.x API Reference*.

Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData, String
keyId) {

       try {
            DecryptRequest decryptRequest = DecryptRequest.builder()
                .ciphertextBlob(encryptedData)
                .keyId(keyId)
                .build();

           DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
           decryptResponse.plaintext();

       } catch (KmsException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see Decrypt in *AWS SDK for Java 2.x API Reference*.

Describe a key

The following code example shows how to describe a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeSpecifcKey(KmsClient kmsClient,String keyId ){

        try {
            DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
                .keyId(keyId)
                .build();

            DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
            System.out.println("The key description is
"+response.keyMetadata().description());
            System.out.println("The key ARN is "+response.keyMetadata().arn());

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
       }
    }
```

- For API details, see DescribeKey in *AWS SDK for Java 2.x API Reference*.

Disable a key

The following code example shows how to disable a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void disableKey( KmsClient kmsClient, String keyId) {

        try {
            DisableKeyRequest keyRequest = DisableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.disableKey(keyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
```

- For API details, see DisableKey in *AWS SDK for Java 2.x API Reference*.

Enable a key

The following code example shows how to enable a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void enableKey(KmsClient kmsClient, String keyId) {

        try {
            EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.enableKey(enableKeyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
  }
```

- For API details, see EnableKey in *AWS SDK for Java 2.x API Reference*.

Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
     public static SdkBytes encryptData(KmsClient kmsClient, String keyId) {

         try {
             SdkBytes myBytes = SdkBytes.fromByteArray(new byte[]{1, 2, 3, 4, 5, 6, 7,
 8, 9, 0});

             EncryptRequest encryptRequest = EncryptRequest.builder()
                 .keyId(keyId)
                 .plaintext(myBytes)
                 .build();

             EncryptResponse response = kmsClient.encrypt(encryptRequest);
             String algorithm = response.encryptionAlgorithm().toString();
             System.out.println("The encryption algorithm is " + algorithm);

             // Get the encrypted data.
```

```
            SdkBytes encryptedData = response.ciphertextBlob();
            return encryptedData;

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

- For API details, see Encrypt in *AWS SDK for Java 2.x API Reference*.

List aliases for a key

The following code example shows how to list aliases for a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllAliases( KmsClient kmsClient) {

        try {
            ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
                .limit(15)
                .build();

            ListAliasesResponse aliasesResponse =
kmsClient.listAliases(aliasesRequest) ;
            List<AliasListEntry> aliases = aliasesResponse.aliases();
            for (AliasListEntry alias: aliases) {
                System.out.println("The alias name is: "+alias.aliasName());
            }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListAliases in *AWS SDK for Java 2.x API Reference*.

List grants for a key

The following code example shows how to list grants for a KMS key.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void displayGrantIds(KmsClient kmsClient, String keyId) {

        try {
```

```
            ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
                .keyId(keyId)
                .limit(15)
                .build();

            ListGrantsResponse response = kmsClient.listGrants(grantsRequest);
            List<GrantListEntry> grants = response.grants();
            for ( GrantListEntry grant: grants) {
                System.out.println("The grant Id is : "+grant.grantId());
            }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListGrants in *AWS SDK for Java 2.x API Reference*.

List keys

The following code example shows how to list KMS keys.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllKeys(KmsClient kmsClient) {

        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();

            ListKeysResponse keysResponse = kmsClient.listKeys(listKeysRequest);
            List<KeyListEntry> keyListEntries = keysResponse.keys();
            for (KeyListEntry key : keyListEntries) {
                System.out.println("The key ARN is: " + key.keyArn());
                System.out.println("The key Id is: " + key.keyId());
            }

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListKeys in *AWS SDK for Java 2.x API Reference*.

Amazon Keyspaces examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Keyspaces.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Amazon Keyspaces

The following code examples show how to get started using Amazon Keyspaces.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
 public class HelloKeyspaces {

    public static void main(String[]args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        listKeyspaces(keyClient);
    }
    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                .maxResults(10)
                .build();

            ListKeyspacesResponse response = keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace: keyspaces) {
                System.out.println("The name of the keyspace is
 "+keyspace.keyspaceName());
            }

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

- For API details, see ListKeyspaces in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions (p. 224)

- Scenarios (p. 238)

Actions

Create a keyspace

The following code example shows how to create an Amazon Keyspaces keyspace.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createKeySpace(KeyspacesClient keyClient, String keyspaceName) {
       try {
           CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
               .keyspaceName(keyspaceName)
               .build();

           CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
           System.out.println("The ARN of the KeySpace is "+response.resourceArn());

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

- For API details, see CreateKeyspace in *AWS SDK for Java 2.x API Reference*.

Create a table

The following code example shows how to create an Amazon Keyspaces table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createTable(KeyspacesClient keyClient, String keySpace, String
tableName) {
       try {
           // Set the columns.
           ColumnDefinition defTitle = ColumnDefinition.builder()
               .name("title")
               .type("text")
               .build();

           ColumnDefinition defYear = ColumnDefinition.builder()
               .name("year")
               .type("int")
               .build();

           ColumnDefinition defReleaseDate = ColumnDefinition.builder()
               .name("release_date")
               .type("timestamp")
```

```
                .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                .name("plot")
                .type("text")
                .build();

            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                .name("year")
                .build();

            PartitionKey titleKey = PartitionKey.builder()
                .name("title")
                .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                .partitionKeys(keyList)
                .allColumns(colList)
                .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                .status(PointInTimeRecoveryStatus.ENABLED)
                .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .schemaDefinition(schemaDefinition)
                .pointInTimeRecovery(timeRecovery)
                .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is "+response.resourceArn());

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

- For API details, see CreateTable in *AWS SDK for Java 2.x API Reference*.

Delete a keyspace

The following code example shows how to delete an Amazon Keyspaces keyspace.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteKeyspace(KeyspacesClient keyClient, String keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

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

- For API details, see DeleteKeyspace in *AWS SDK for Java 2.x API Reference*.

Delete a table

The following code example shows how to delete an Amazon Keyspaces table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName){
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            keyClient.deleteTable(tableRequest);

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

- For API details, see DeleteTable in *AWS SDK for Java 2.x API Reference*.

Get data about a keyspace

The following code example shows how to get data about an Amazon Keyspaces keyspace.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
```

```
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The "+ name+ " KeySpace is ready");

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

- For API details, see GetKeyspace in *AWS SDK for Java 2.x API Reference*.

Get data about a table

The following code example shows how to get data about an Amazon Keyspaces table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is "+status);

                if (status.compareTo("ACTIVE") ==0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def: cols) {
                System.out.println("The column name is "+def.name());
                System.out.println("The column type is "+def.type());
            }

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

- For API details, see GetTable in *AWS SDK for Java 2.x API Reference*.

List keyspaces

The following code example shows how to list Amazon Keyspaces keyspaces.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                .maxResults(10)
                .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                .flatMap(r -> r.keyspaces().stream())
                .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

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

- For API details, see ListKeyspaces in *AWS SDK for Java 2.x API Reference*.

List tables in a keyspace

The following code example shows how to list Amazon Keyspaces tables in a keyspace.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            ListTablesIterable listRes = keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                .flatMap(r -> r.tables().stream())
                .forEach(content -> System.out.println(" ARN: " + content.resourceArn()
+
                    " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

- For API details, see ListTables in *AWS SDK for Java 2.x API Reference*.

Restore a table to a point in time

The following code example shows how to restore an Amazon Keyspaces table to a point in time.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
       try {
           Instant myTime = utc.toInstant();
           RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
               .restoreTimestamp(myTime)
               .sourceTableName("Movie")
               .targetKeyspaceName(keyspaceName)
               .targetTableName("MovieRestore")
               .sourceKeyspaceName(keyspaceName)
               .build();

           RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
           System.out.println("The ARN of the restored table is
"+response.restoredTableARN());

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

- For API details, see RestoreTable in *AWS SDK for Java 2.x API Reference*.

Update a table

The following code example shows how to update an Amazon Keyspaces table.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void updateTable(KeyspacesClient keyClient, String keySpace, String
tableName){
       try {
           ColumnDefinition def = ColumnDefinition.builder()
               .name("watched")
```

```
                .type("boolean")
                .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .addColumns(def)
                .build();

            keyClient.updateTable(tableRequest);

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

- For API details, see UpdateTable in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with keyspaces and tables

The following code example shows how to:

- Create a keyspace and table. The table schema holds movie data and has point-in-time recovery enabled.
- Connect to the keyspace using a secure TLS connection with SigV4 authentication.
- Query the table. Add, retrieve, and update movie data.
- Update the table. Add a column to track watched movies.
- Restore the table to its previous state and clean up resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
```

```
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-") ;

    /*
    Usage:
      fileName - The name of the JSON file that contains movie data. (Get this file
 from the GitHub repo at resources/sample_file.)
      keyspaceName - The name of the keyspace to create.
   */
    public static void main(String[]args) throws InterruptedException, IOException {
        String fileName = "<Replace with the JSON file that contains movie data>" ;
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013 ;
        String tableName = "Movie" ;
        String tableNameRestore = "MovieRestore" ;
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        DriverConfigLoader loader =
 DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
            .withConfigLoader(loader)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a keyspace.");
        createKeySpace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        Thread.sleep(5000);
        System.out.println("2. Check for keyspace existence.");
        checkKeyspaceExistence(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into the
Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using the
timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String keyspaceName,
String tableName)throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
            GetTableRequest tableRequest = GetTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            // Keep looping until table cannot be found and a ResourceNotFoundException
is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is "+status);
                Thread.sleep(500);
            }

        } catch (ResourceNotFoundException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println("The table is deleted");
```

```
    }

    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
 String tableName){
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRestoredTable(KeyspacesClient keyClient, String
 keyspaceName, String tableName)throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println("The table status is "+status);

                if (status.compareTo("ACTIVE") ==0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def: cols) {
                System.out.println("The column name is "+def.name());
                System.out.println("The column type is "+def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
 ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                .restoreTimestamp(myTime)
                .sourceTableName("Movie")
                .targetKeyspaceName(keyspaceName)
                .targetTableName("MovieRestore")
                .sourceKeyspaceName(keyspaceName)
                .build();

            RestoreTableResponse response =
 keyClient.restoreTable(restoreTableRequest);
```

```
            System.out.println("The ARN of the restored table is
 "+response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \""+keyspaceName+"\".
\"Movie\" WHERE watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
 titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \""+keySpace+"\".\"Movie\" SET watched=true WHERE
 title = :k0 AND year = :k1;";
        BatchStatementBuilder builder =
 BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace, String
 tableName){
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                .name("watched")
                .type("boolean")
                .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .addColumns(def)
                .build();

            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificMovie(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \""+keyspaceName+"\".
\"Movie\" WHERE title = 'The Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
```

```
    }

    // Get records from the Movie table.
    public static void getMovieData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \""+keyspaceName+"\".
\"Movie\";");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String keySpace)
 throws IOException {
        String sqlStatement = "INSERT INTO \""+keySpace +"\".\"Movie\" (title, year,
 plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0 ;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
                break ;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.
            BatchStatementBuilder builder =
 BatchStatement.builder(DefaultBatchType.UNLOGGED);
            builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
            PreparedStatement preparedStatement = session.prepare(sqlStatement);
            builder.addStatement(preparedStatement.boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", plot)
                .build());

            BatchStatement batchStatement = builder.build();
            session.execute(batchStatement);
            t++;
        }

        System.out.println("You have added " +t +" records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            ListTablesIterable listRes = keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                .flatMap(r -> r.tables().stream())
                .forEach(content -> System.out.println(" ARN: " + content.resourceArn()
 +
                    " Table name: " + content.tableName()));
```

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

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
 String tableName)throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is "+status);

                if (status.compareTo("ACTIVE") ==0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def: cols) {
                System.out.println("The column name is "+def.name());
                System.out.println("The column type is "+def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createTable(KeyspacesClient keyClient, String keySpace, String
 tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
                .name("title")
                .type("text")
                .build();

            ColumnDefinition defYear = ColumnDefinition.builder()
                .name("year")
                .type("int")
                .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                .name("release_date")
                .type("timestamp")
                .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                .name("plot")
                .type("text")
                .build();

            List<ColumnDefinition> colList = new ArrayList<>();
```

```
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                .name("year")
                .build();

            PartitionKey titleKey = PartitionKey.builder()
                .name("title")
                .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                .partitionKeys(keyList)
                .allColumns(colList)
                .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                .status(PointInTimeRecoveryStatus.ENABLED)
                .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .schemaDefinition(schemaDefinition)
                .pointInTimeRecovery(timeRecovery)
                .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is "+response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                .maxResults(10)
                .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                .flatMap(r -> r.keyspaces().stream())
                .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
```

```
                .keyspaceName(keyspaceName)
                .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The "+ name+ " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            CreateKeyspaceResponse response =
 keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is "+response.resourceArn());

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Kinesis examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Kinesis.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a stream

The following code example shows how to create a Kinesis stream.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void createStream(KinesisClient kinesisClient, String streamName) {

    try {
        CreateStreamRequest streamReq = CreateStreamRequest.builder()
            .streamName(streamName)
            .shardCount(1)
            .build();

        kinesisClient.createStream(streamReq);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see CreateStream in *AWS SDK for Java 2.x API Reference*.

Delete a stream

The following code example shows how to delete a Kinesis stream.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void deleteStream(KinesisClient kinesisClient, String streamName) {

    try {
        DeleteStreamRequest delStream = DeleteStreamRequest.builder()
            .streamName(streamName)
            .build();

        kinesisClient.deleteStream(delStream);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see DeleteStream in *AWS SDK for Java 2.x API Reference*.

Get data in batches from a stream

The following code example shows how to get data in batches from a Kinesis stream.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getStockTrades(KinesisClient kinesisClient, String streamName) {

       String shardIterator;
       String lastShardId = null;

       // Retrieve the Shards from a Stream
       DescribeStreamRequest describeStreamRequest = DescribeStreamRequest.builder()
           .streamName(streamName)
           .build();

       List<Shard> shards = new ArrayList<>();
       DescribeStreamResponse streamRes;
       do {
           streamRes = kinesisClient.describeStream(describeStreamRequest);
           shards.addAll(streamRes.streamDescription().shards());

           if (shards.size() > 0) {
               lastShardId = shards.get(shards.size() - 1).shardId();
           }
           } while (streamRes.streamDescription().hasMoreShards());

       GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
           .streamName(streamName)
           .shardIteratorType("TRIM_HORIZON")
           .shardId(lastShardId)
           .build();

       GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
       shardIterator = shardIteratorResult.shardIterator();

       // Continuously read data records from shard.
       List<Record> records;

       // Create new GetRecordsRequest with existing shardIterator.
       // Set maximum records to return to 1000.
       GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
           .shardIterator(shardIterator)
           .limit(1000)
           .build();

       GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

       // Put result into record list. Result may be empty.
       records = result.records();

       // Print records
       for (Record record : records) {
           SdkBytes byteBuffer = record.data();
           System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
       }
   }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - GetRecords
  - GetShardIterator

Put data into a stream

The following code example shows how to put data into a Kinesis stream.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void setStockData( KinesisClient kinesisClient, String streamName) {

        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x=0; x<index; x++){
                StockTrade trade = stockTradeGenerator.getRandomTrade();
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }

    private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
                                       String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization by
 the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
            .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as the
 partition key, explained in the Supplemental Information section below.
            .streamName(streamName)
            .data(SdkBytes.fromByteArray(bytes))
            .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }
```

```
    private static void validateStream(KinesisClient kinesisClient, String streamName)
 {
        try {
            DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if(!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
 {
                System.err.println("Stream " + streamName + " is not active. Please
 wait a few moments and try again.");
                System.exit(1);
            }

        }catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
 streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
```

- For API details, see PutRecord in *AWS SDK for Java 2.x API Reference*.

Lambda examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Lambda.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a function

The following code example shows how to create a Lambda function.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createLambdaFunction(LambdaClient awsLambda,
                                           String functionName,
                                           String filePath,
                                           String role,
                                           String handler) {

       try {
           LambdaWaiter waiter = awsLambda.waiter();
           InputStream is = new FileInputStream(filePath);
           SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

           FunctionCode code = FunctionCode.builder()
               .zipFile(fileToUpload)
               .build();

           CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
               .functionName(functionName)
               .description("Created by the Lambda Java API")
               .code(code)
               .handler(handler)
               .runtime(Runtime.JAVA8)
               .role(role)
               .build();

           // Create a Lambda function using a waiter.
           CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
           GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
               .functionName(functionName)
               .build();
           WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("The function ARN is " +
functionResponse.functionArn());

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

- For API details, see CreateFunction in *AWS SDK for Java 2.x API Reference*.

Delete a function

The following code example shows how to delete a Lambda function.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
       try {
           DeleteFunctionRequest request = DeleteFunctionRequest.builder()
               .functionName(functionName)
               .build();
```

```
            awsLambda.deleteFunction(request);
            System.out.println("The "+functionName +" function was deleted");

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteFunction in *AWS SDK for Java 2.x API Reference*.

Invoke a function

The following code example shows how to invoke a Lambda function.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {

        InvokeResponse res = null ;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json) ;

            // Setup an InvokeRequest.
            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String() ;
            System.out.println(value);

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see Invoke in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with functions

The following code example shows how to:

- Create an IAM role and Lambda function, then upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure with an environment variable.

- Invoke the function with new parameters and get results. Display the returned execution log.
- List the functions for your account, then clean up resources.

For more information, see Create a Lambda function with the console.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/*
 *  Lambda function names appear as:
 *
 *  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 *  To find this value, look at the function in the AWS Management Console.
 *
 *  Before running this Java code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see this documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <functionName> <filePath> <role> <handler> <bucketName> <key> \n\n" +
            "Where:\n" +
            "    functionName - The name of the Lambda function. \n"+
            "    filePath - The path to the .zip or .jar where the code is located.
 \n"+
            "    role - The AWS Identity and Access Management (IAM) service role that
 has Lambda permissions. \n"+
            "    handler - The fully qualified method name (for example,
 example.Handler::handleRequest). \n"+
            "    bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
 that contains the .zip or .jar used to update the Lambda function's code. \n"+
            "    key - The Amazon S3 key name that represents the .zip or .jar (for
 example, LambdaHello-1.0-SNAPSHOT.jar)." ;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
```

```
        String role = args[2];
        String handler = args[3];
        String bucketName = args[4];
        String key = args[5];

        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Lambda example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
        String funArn = createLambdaFunction(awsLambda, functionName, filePath, role,
handler);
        System.out.println("The AWS Lambda ARN is "+funArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get the "+functionName + " AWS Lambda function.");
        getFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List all AWS Lambda functions.");
        listFunctions(awsLambda);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Invoke the Lambda function.");
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String filePath,
```

```
                                            String role,
                                            String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA8)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

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

    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is "
+response.configuration().runtime());

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config: list) {
                System.out.println("The function name is "+config.functionName());
            }

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    public static void invokeFunction(LambdaClient awsLambda, String functionName) {

        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json) ;

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String() ;
            System.out.println(value);

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateFunctionCode(LambdaClient awsLambda, String functionName,
String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                .functionName(functionName)
                .publish(true)
                .s3Bucket(bucketName)
                .s3Key(key)
                .build();

            UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest) ;
            GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter.waitUntilFunctionUpdated(getFunctionConfigRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +response.lastModified());

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler ){
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA11 )
```

```
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName ) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The "+functionName +" function was deleted");

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

MediaConvert examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with MediaConvert.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a transcoding job

The following code example shows how to create a MediaConvert transcoding job.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
 software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
```

```
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
 software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
 software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon Resource Name
 (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class CreateJob {

   public static void main(String[] args) {

        final String usage = "\n" +
                "Usage:\n" +
```

```
                "    <mcRoleARN> <fileInput> \n\n" +
                "Where:\n" +
                "    mcRoleARN - The MediaConvert Role ARN. \n"+
                "    fileInput -  The URL of an Amazon S3 bucket where the input file
is located. \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String mcRoleARN = args[0];
        String fileInput = args[1];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

       String id = createMediaJob(mc, mcRoleARN, fileInput);
       System.out.println("MediaConvert job created. Job Id = " +id );
       mc.close();
   }

   public static String createMediaJob(MediaConvertClient mc, String mcRoleARN, String
fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') + 1) +
"javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
        String mp4Output = s3path + "mp4/";

        try {
            DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint URL!");
                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
            System.out.println("MediaConvert service URL: " + endpointURL);
            System.out.println("MediaConvert role arn: " + mcRoleARN);
            System.out.println("MediaConvert input file: " + fileInput);
            System.out.println("MediaConvert output path: " + s3path);

            MediaConvertClient emc = MediaConvertClient.builder()
                    .region(Region.US_WEST_2)
                    .endpointOverride(URI.create(endpointURL))
                    .build();

            // output group Preset HLS low profile
            Output hlsLow = createOutput("hls_low", "_low", "_$dt$", 750000, 7, 1920,
1080, 640);
            // output group Preset HLS media profile
            Output hlsMedium = createOutput("hls_medium", "_medium", "_$dt$", 1200000,
7, 1920, 1080, 1280);
            // output group Preset HLS high profole
            Output hlsHigh = createOutput("hls_high", "_high", "_$dt$", 3500000, 8,
1920, 1080, 1920);

            OutputGroup appleHLS = OutputGroup.builder().name("Apple
HLS").customName("Example")
```

```
.outputGroupSettings(OutputGroupSettings.builder().type(OutputGroupType.HLS_GROUP_SETTINGS)
                            .hlsGroupSettings(HlsGroupSettings.builder()

.directoryStructure(HlsDirectoryStructure.SINGLE_DIRECTORY)

.manifestDurationFormat(HlsManifestDurationFormat.INTEGER)

.streamInfResolution(HlsStreamInfResolution.INCLUDE)
                                    .clientCache(HlsClientCache.ENABLED)

.captionLanguageSetting(HlsCaptionLanguageSetting.OMIT)
                                    .manifestCompression(HlsManifestCompression.NONE)
                                    .codecSpecification(HlsCodecSpecification.RFC_4281)

.outputSelection(HlsOutputSelection.MANIFESTS_AND_SEGMENTS)

.programDateTime(HlsProgramDateTime.EXCLUDE).programDateTimePeriod(600)

.timedMetadataId3Frame(HlsTimedMetadataId3Frame.PRIV).timedMetadataId3Period(10)

.destination(fileOutput).segmentControl(HlsSegmentControl.SEGMENTED_FILES)
                                    .minFinalSegmentLength((double)
0).segmentLength(4).minSegmentLength(0).build())
                            .build())
                    .outputs(hlsLow, hlsMedium, hlsHigh).build();

           OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder().type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder().destination(mp4Output).build()).build())
                    .outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder().container(ContainerType.MP4).build())

.videoDescription(VideoDescription.builder().width(1280).height(720)

.scalingBehavior(ScalingBehavior.DEFAULT).sharpness(50).antiAlias(AntiAlias.ENABLED)
                                    .timecodeInsertion(VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT).respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE).dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder().codec(VideoCodec.H_264)
                                            .h264Settings(H264Settings.builder()

.rateControlMode(H264RateControlMode.QVBR)

.parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

.qualityTuningLevel(H264QualityTuningLevel.SINGLE_PASS)
                                                    .qvbrSettings(

H264QvbrSettings.builder().qvbrQualityLevel(8).build())

.codecLevel(H264CodecLevel.AUTO).codecProfile(H264CodecProfile.MAIN)
                                                    .maxBitrate(2400000)

.framerateControl(H264FramerateControl.INITIALIZE_FROM_SOURCE)

.gopSize(2.0).gopSizeUnits(H264GopSizeUnits.SECONDS)

.numberBFramesBetweenReferenceFrames(2).gopClosedCadence(1)
```

```
.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED).syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(3).dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(H264SceneChangeDetect.ENABLED).minIInterval(0)
                                                    .telecine(H264Telecine.NONE)
                                                    .framerateConversionAlgorithm(

H264FramerateConversionAlgorithm.DUPLICATE_DROP)

.entropyEncoding(H264EntropyEncoding.CABAC).slices(1)

.unregisteredSeiTimecode(H264UnregisteredSeiTimecode.DISABLED)
                                                    .repeatPps(H264RepeatPps.DISABLED)

.adaptiveQuantization(H264AdaptiveQuantization.HIGH)
                                                    .spatialAdaptiveQuantization(

H264SpatialAdaptiveQuantization.ENABLED)
                                                    .temporalAdaptiveQuantization(

H264TemporalAdaptiveQuantization.ENABLED)
                                                    .flickerAdaptiveQuantization(

H264FlickerAdaptiveQuantization.DISABLED)

.softness(0).interlaceMode(H264InterlaceMode.PROGRESSIVE).build())
                                            .build())
                                    .build())
                            .audioDescriptions(AudioDescription.builder()
                                    .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

.languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

.codecSettings(AudioCodecSettings.builder().codec(AudioCodec.AAC)

.aacSettings(AacSettings.builder().codecProfile(AacCodecProfile.LC)

.rateControlMode(AacRateControlMode.CBR)

.codingMode(AacCodingMode.CODING_MODE_2_0).sampleRate(44100)

.bitrate(160000).rawFormat(AacRawFormat.NONE)

.specification(AacSpecification.MPEG4)
                                                    .audioDescriptionBroadcasterMix(

AacAudioDescriptionBroadcasterMix.NORMAL)
                                                    .build())
                                            .build())
                                    .build())
                            .build())
                    .build();
            OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder().type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder().destination(thumbsOutput).build()).build())
                    .outputs(Output.builder().extension("jpg")
```

```
.containerSettings(ContainerSettings.builder().container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder().scalingBehavior(ScalingBehavior.DEFAULT)
                                  .sharpness(50).antiAlias(AntiAlias.ENABLED)
                                  .timecodeInsertion(VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT).dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder().codec(VideoCodec.FRAME_CAPTURE)

.frameCaptureSettings(FrameCaptureSettings.builder().framerateNumerator(1)

.framerateDenominator(1).maxCaptures(10000000).quality(80).build())
                                          .build())
                                  .build())
                          .build())
                  .build();

           Map<String, AudioSelector> audioSelectors = new HashMap<>();
           audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT).offset(0).build());

           JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder().audioSelectors(audioSelectors)
                  .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW).rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0).deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                  .outputGroups(appleHLS, thumbs, fileMp4).build();

           CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN).settings(jobSettings)
                  .build();

          CreateJobResponse createJobResponse = emc.createJob(createJobRequest);
         return createJobResponse.job().id();

       } catch (MediaConvertException e) {
           System.out.println(e.toString());
           System.exit(0);
       }
       return "";
   }

   private final static Output createOutput(String customName,
                                            String nameModifier,
                                            String segmentModifier,
                                            int qvbrMaxBitrate,
                                            int qvbrQualityLevel,
                                            int originWidth,
                                            int originHeight,
                                            int targetWidth) {

       int targetHeight = Math.round(originHeight * targetWidth / originWidth)
               - (Math.round(originHeight * targetWidth / originWidth) % 4);
       Output output = null;
       try {
           output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()
```

```
.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier).audioGroupId("program_audio")
                           .iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                    .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)
                           .m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET).pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0).scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE).timedMetadata(TimedMetadata.NONE)
                                   .timedMetadataPid(502).videoPid(481)
                                   .audioPids(482, 483, 484, 485, 486, 487, 488, 489,
490, 491, 492).build())
                           .build())
                    .videoDescription(

VideoDescription.builder().width(targetWidth).height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT).sharpness(50).antiAlias(AntiAlias.ENABLED)
                                   .timecodeInsertion(VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT).respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE).dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder().codec(VideoCodec.H_264)
                                            .h264Settings(H264Settings.builder()

.rateControlMode(H264RateControlMode.QVBR)

.parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

.qualityTuningLevel(H264QualityTuningLevel.SINGLE_PASS)

.qvbrSettings(H264QvbrSettings.builder()

.qvbrQualityLevel(qvbrQualityLevel).build())
                                                        .codecLevel(H264CodecLevel.AUTO)
                                                        .codecProfile((targetHeight > 720
&& targetWidth > 1280)
                                                                ? H264CodecProfile.HIGH
                                                                : H264CodecProfile.MAIN)
                                                        .maxBitrate(qvbrMaxBitrate)

.framerateControl(H264FramerateControl.INITIALIZE_FROM_SOURCE)

.gopSize(2.0).gopSizeUnits(H264GopSizeUnits.SECONDS)

.numberBFramesBetweenReferenceFrames(2).gopClosedCadence(1)

.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED).syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(3).dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(H264SceneChangeDetect.ENABLED).minIInterval(0)
                                                        .telecine(H264Telecine.NONE)
                                                        .framerateConversionAlgorithm(

H264FramerateConversionAlgorithm.DUPLICATE_DROP)
```

```
 .entropyEncoding(H264EntropyEncoding.CABAC).slices(1)

 .unregisteredSeiTimecode(H264UnregisteredSeiTimecode.DISABLED)
                                                     .repeatPps(H264RepeatPps.DISABLED)

 .adaptiveQuantization(H264AdaptiveQuantization.HIGH)
                                                     .spatialAdaptiveQuantization(

 H264SpatialAdaptiveQuantization.ENABLED)
                                                     .temporalAdaptiveQuantization(

 H264TemporalAdaptiveQuantization.ENABLED)
                                                     .flickerAdaptiveQuantization(

 H264FlickerAdaptiveQuantization.DISABLED)

 .softness(0).interlaceMode(H264InterlaceMode.PROGRESSIVE).build())
                                          .build())
                                   .build())

 .audioDescriptions(AudioDescription.builder().audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
                            .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

 .codecSettings(AudioCodecSettings.builder().codec(AudioCodec.AAC).aacSettings(AacSettings

 .builder().codecProfile(AacCodecProfile.LC).rateControlMode(AacRateControlMode.CBR)

 .codingMode(AacCodingMode.CODING_MODE_2_0).sampleRate(44100).bitrate(96000)

 .rawFormat(AacRawFormat.NONE).specification(AacSpecification.MPEG4)

 .audioDescriptionBroadcasterMix(AacAudioDescriptionBroadcasterMix.NORMAL).build())
                                   .build())
                            .build())
                    .build();
        } catch (MediaConvertException e) {
            e.printStackTrace();
            System.exit(0);
        }
        return output;
    }
}
```

- For API details, see CreateJob in *AWS SDK for Java 2.x API Reference*.

Get a transcoding job

The following code example shows how to get a MediaConvert transcoding job.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getSpecificJob(MediaConvertClient mc, String jobId) {

        try {
            DescribeEndpointsResponse res =
 mc.describeEndpoints(DescribeEndpointsRequest.builder()
```

```
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint URL!");
                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            GetJobRequest jobRequest = GetJobRequest.builder()
                .id(jobId)
                .build();

            GetJobResponse response = emc.getJob(jobRequest);
            System.out.println("The ARN of the job is "+response.job().arn());

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
```

- For API details, see GetJob in *AWS SDK for Java 2.x API Reference*.

List transcoding jobs

The following code example shows how to list MediaConvert transcoding jobs.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listCompleteJobs(MediaConvertClient mc) {

        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint URL!");
                System.exit(1);
            }

            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            ListJobsRequest jobsRequest = ListJobsRequest.builder()
                .maxResults(10)
                .status("COMPLETE")
                .build();
```

```
            ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job: jobs) {
                System.out.println("The JOB ARN is : "+job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
```

- For API details, see ListJobs in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a batch interface job

The following code example shows how to create a Amazon Personalize batch interface job.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                                                            String solutionVersionArn,
                                                            String jobName,
                                                            String
s3InputDataSourcePath,
                                                            String
s3DataDestinationPath,
                                                            String roleArn,
                                                            String explorationWeight,
                                                            String
explorationItemAgeCutOff) {

       long waitInMilliseconds = 60 * 1000;
       String status;
       String batchInferenceJobArn;
```

```
        try {

            // Set up data input and output parameters.
            S3DataConfig inputSource = S3DataConfig.builder()
                .path(s3InputDataSourcePath)
                .build();

            S3DataConfig outputDestination = S3DataConfig.builder()
                .path(s3DataDestinationPath)
                .build();

            BatchInferenceJobInput jobInput = BatchInferenceJobInput.builder()
                .s3DataSource(inputSource)
                .build();

            BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
                .s3DataDestination(outputDestination)
                .build();

            // Optional code to build the User-Personalization specific item
exploration config.
            HashMap<String, String> explorationConfig = new HashMap<>();

            explorationConfig.put("explorationWeight", explorationWeight);
            explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

            BatchInferenceJobConfig jobConfig = BatchInferenceJobConfig.builder()
                .itemExplorationConfig(explorationConfig)
                .build();

            // End optional User-Personalization recipe specific code.

            CreateBatchInferenceJobRequest createBatchInferenceJobRequest =
CreateBatchInferenceJobRequest.builder()
                .solutionVersionArn(solutionVersionArn)
                .jobInput(jobInput)
                .jobOutput(jobOutputLocation)
                .jobName(jobName)
                .roleArn(roleArn)
                .batchInferenceJobConfig(jobConfig)   // Optional
                .build();

            batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                 .batchInferenceJobArn();

            DescribeBatchInferenceJobRequest describeBatchInferenceJobRequest =
DescribeBatchInferenceJobRequest.builder()
                .batchInferenceJobArn(batchInferenceJobArn)
                .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
            while (Instant.now().getEpochSecond() < maxTime) {

                BatchInferenceJob batchInferenceJob = personalizeClient
                    .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                    .batchInferenceJob();

                status = batchInferenceJob.status();
                System.out.println("Batch inference job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
```

```
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return batchInferenceJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- For API details, see CreateBatchInferenceJob in *AWS SDK for Java 2.x API Reference*.

Create a campaign

The following code example shows how to create a Amazon Personalize campaign.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createPersonalCompaign(PersonalizeClient personalizeClient,
String solutionVersionArn, String name) {

       try {
           CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
               .minProvisionedTPS(1)
               .solutionVersionArn(solutionVersionArn)
               .name(name)
               .build();

           CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
           System.out.println("The campaign ARN is "+campaignResponse.campaignArn());

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

- For API details, see CreateCampaign in *AWS SDK for Java 2.x API Reference*.

Create a dataset

The following code example shows how to create a Amazon Personalize dataset.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createDataset(PersonalizeClient personalizeClient,
                                       String datasetName,
                                       String datasetGroupArn,
                                       String datasetType,
                                       String schemaArn) {
        try {
            CreateDatasetRequest request = CreateDatasetRequest.builder()
                .name(datasetName)
                .datasetGroupArn(datasetGroupArn)
                .datasetType(datasetType)
                .schemaArn(schemaArn)
                .build();

            String datasetArn = personalizeClient.createDataset(request)
                .datasetArn();
            System.out.println("Dataset " + datasetName + " created.");
            return datasetArn;

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

- For API details, see CreateDataset in *AWS SDK for Java 2.x API Reference*.

Create a dataset export job

The following code example shows how to create a Amazon Personalize dataset export job.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createDatasetExportJob(PersonalizeClient personalizeClient,
                                                String jobName,
                                                String datasetArn,
                                                IngestionMode ingestionMode,
                                                String roleArn,
                                                String s3BucketPath,
                                                String kmsKeyArn) {

        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String status = null;

        try {

            S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
            DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig).build();

            CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
                    .jobName(jobName)
                    .datasetArn(datasetArn)
```

```
                    .ingestionMode(ingestionMode)
                    .jobOutput(jobOutput)
                    .roleArn(roleArn)
                    .build();

            String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

            DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
                .datasetExportJobArn(datasetExportJobArn)
                .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetExportJob datasetExportJob =
personalizeClient.describeDatasetExportJob(describeDatasetExportJobRequest)
                        .datasetExportJob();

                status = datasetExportJob.status();
                System.out.println("Export job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                   return status;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- For API details, see CreateDatasetExportJob in *AWS SDK for Java 2.x API Reference*.

Create a dataset group

The following code example shows how to create a Amazon Personalize dataset group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createDatasetGroup(PersonalizeClient personalizeClient, String
datasetGroupName) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .build();
            return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
```

```
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

Create a domain dataset group.

```
    public static String createDomainDatasetGroup(PersonalizeClient personalizeClient,
                                                  String datasetGroupName,
                                                  String domain) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .domain(domain)
                    .build();
            return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- For API details, see CreateDatasetGroup in *AWS SDK for Java 2.x API Reference*.

Create a dataset import job

The following code example shows how to create a Amazon Personalize dataset import job.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
                                                           String jobName,
                                                           String datasetArn,
                                                           String s3BucketPath,
                                                           String roleArn) {

        long waitInMilliseconds = 60 * 1000;
        String status;
        String datasetImportJobArn;

        try {
            DataSource importDataSource = DataSource.builder()
                    .dataLocation(s3BucketPath)
                    .build();

            CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                    .datasetArn(datasetArn)
                    .dataSource(importDataSource)
                    .jobName(jobName)
```

```
                    .roleArn(roleArn)
                    .build();

            datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                    .datasetImportJobArn();
            DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                    .datasetImportJobArn(datasetImportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetImportJob datasetImportJob = personalizeClient
                        .describeDatasetImportJob(describeDatasetImportJobRequest)
                        .datasetImportJob();

                status = datasetImportJob.status();
                System.out.println("Dataset import job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return datasetImportJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- For API details, see CreateDatasetImportJob in *AWS SDK for Java 2.x API Reference*.

Create a domain schema

The following code example shows how to create a Amazon Personalize domain schema.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createDomainSchema(PersonalizeClient personalizeClient, String
schemaName, String domain, String filePath) {

        String schema = null;
        try {
            schema = new String(Files.readAllBytes(Paths.get(filePath)));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
```

```
        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .domain(domain)
                    .schema(schema)
                    .build();

            String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;

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

- For API details, see CreateSchema in *AWS SDK for Java 2.x API Reference*.

Create a filter

The following code example shows how to create a Amazon Personalize filter.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createFilter(PersonalizeClient personalizeClient,
                                      String filterName,
                                      String datasetGroupArn,
                                      String filterExpression) {
        try {
            CreateFilterRequest request = CreateFilterRequest.builder()
                    .name(filterName)
                    .datasetGroupArn(datasetGroupArn)
                    .filterExpression(filterExpression)
                    .build();

            return personalizeClient.createFilter(request).filterArn();
        }
        catch(PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateFilter in *AWS SDK for Java 2.x API Reference*.

Create a recommender

The following code example shows how to create a Amazon Personalize recommender.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createRecommender(PersonalizeClient personalizeClient,
                                          String name,
                                          String datasetGroupArn,
                                          String recipeArn) {

       long maxTime = 0;
       long waitInMilliseconds = 30 * 1000; // 30 seconds
       String recommenderStatus = "";

       try {
               CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                       .datasetGroupArn(datasetGroupArn)
                       .name(name)
                       .recipeArn(recipeArn)
                       .build();

               CreateRecommenderResponse recommenderResponse =
personalizeClient.createRecommender(createRecommenderRequest);
               String recommenderArn = recommenderResponse.recommenderArn();
               System.out.println("The recommender ARN is " + recommenderArn);

               DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                       .recommenderArn(recommenderArn)
                       .build();

               maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

               while (Instant.now().getEpochSecond() < maxTime) {

                   recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender().status();
                   System.out.println("Recommender status: " + recommenderStatus);

                   if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
                       break;
                   }
                   try {
                       Thread.sleep(waitInMilliseconds);
                   } catch (InterruptedException e) {
                       System.out.println(e.getMessage());
                   }
               }
               return recommenderArn;

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

- For API details, see CreateRecommender in *AWS SDK for Java 2.x API Reference*.

Create a schema

The following code example shows how to create a Amazon Personalize schema.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

       String schema = null;
       try {
           schema = new String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }

       try {
           CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                   .name(schemaName)
                   .schema(schema)
                   .build();

           String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

           System.out.println("Schema arn: " + schemaArn);

           return schemaArn;

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

- For API details, see CreateSchema in *AWS SDK for Java 2.x API Reference*.

Create a solution

The following code example shows how to create a Amazon Personalize solution.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createPersonalizeSolution(PersonalizeClient personalizeClient,
                                                  String datasetGroupArn,
                                                  String solutionName,
                                                  String recipeArn) {

       try {
           CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
               .name(solutionName)
```

```
                .datasetGroupArn(datasetGroupArn)
                .recipeArn(recipeArn)
                .build();

            CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
            return solutionResponse.solutionArn();

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

- For API details, see CreateSolution in *AWS SDK for Java 2.x API Reference*.

Create a solution version

The following code example shows how to create a Amazon Personalize solution.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String solutionStatus = "";
        String solutionVersionStatus = "";
        String solutionVersionArn = "";

        try {
            DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
                .solutionArn(solutionArn)
                .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            // Wait until solution is active.
            while (Instant.now().getEpochSecond() < maxTime) {

                solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
                System.out.println("Solution status: " + solutionStatus);

                if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
```

```
            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient.createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " + solutionVersionArn);

                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                    .solutionVersionArn(solutionVersionArn)
                    .build();

                while (Instant.now().getEpochSecond() < maxTime) {

                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest).solutionVersion().status
                    System.out.println("Solution version status: " +
solutionVersionStatus);

                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
                    try {
                        Thread.sleep(waitInMilliseconds);
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                return solutionVersionArn;
            }
        } catch(PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see CreateSolutionVersion in *AWS SDK for Java 2.x API Reference*.

Create an event tracker

The following code example shows how to create a Amazon Personalize event tracker.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createEventTracker(PersonalizeClient personalizeClient, String
eventTrackerName, String datasetGroupArn) {

        String eventTrackerId = "";
        String eventTrackerArn;
```

```
        long maxTime = 3 * 60 * 60; // 3 hours
        long waitInMilliseconds = 20 * 1000; // 20 seconds
        String status;

        try {

            CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
                .name(eventTrackerName)
                .datasetGroupArn(datasetGroupArn)
                .build();

            CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient.createEventTracker(createEventTrackerRequest);

            eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
            eventTrackerId = createEventTrackerResponse.trackingId();
            System.out.println("Event tracker ARN: " + eventTrackerArn);
            System.out.println("Event tracker ID: " + eventTrackerId);

            maxTime = Instant.now().getEpochSecond() + maxTime;

            DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
                .eventTrackerArn(eventTrackerArn)
                .build();


            while (Instant.now().getEpochSecond() < maxTime) {

                status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
                System.out.println("EventTracker status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return eventTrackerId;
        }
        catch (PersonalizeException e){
            System.out.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return eventTrackerId;
    }
```

- For API details, see CreateEventTracker in *AWS SDK for Java 2.x API Reference*.

Delete a campaign

The following code example shows how to delete a campaign in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn ) {

       try {
           DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
               .campaignArn(campaignArn)
               .build();

           personalizeClient.deleteCampaign(campaignRequest);

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

- For API details, see DeleteCampaign in *AWS SDK for Java 2.x API Reference*.

Delete a solution

The following code example shows how to delete a solution in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteGivenSolution(PersonalizeClient personalizeClient, String
solutionArn ) {

       try {
           DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
               .solutionArn(solutionArn)
               .build();

           personalizeClient.deleteSolution(solutionRequest);
           System.out.println("Done");

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

- For API details, see DeleteSolution in *AWS SDK for Java 2.x API Reference*.

Delete an event tracker

The following code example shows how to delete an event tracker in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteEventTracker(PersonalizeClient personalizeClient, String
eventTrackerArn) {
       try {
           DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
               .eventTrackerArn(eventTrackerArn)
               .build();

           int status =

personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

           System.out.println("Status code:" + status);

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

- For API details, see DeleteEventTracker in *AWS SDK for Java 2.x API Reference*.

Describe a campaign

The following code example shows how to describe a campaign in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

       try {
           DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
               .campaignArn(campaignArn)
               .build();

           DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
           Campaign myCampaign = campaignResponse.campaign();
           System.out.println("The Campaign name is "+myCampaign.name());
           System.out.println("The Campaign status is "+myCampaign.status());

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

- For API details, see DescribeCampaign in *AWS SDK for Java 2.x API Reference*.

Describe a recipe

The following code example shows how to describe a recipe in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

       try{
           DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
               .recipeArn(recipeArn)
               .build();

           DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
           System.out.println("The recipe name is "+recipeResponse.recipe().name());

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

- For API details, see DescribeRecipe in *AWS SDK for Java 2.x API Reference*.

Describe a solution

The following code example shows how to describe a solution in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
  public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

      try {
          DescribeSolutionRequest solutionRequest = DescribeSolutionRequest.builder()
              .solutionArn(solutionArn)
              .build();

          DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
          System.out.println("The Solution name is "+response.solution().name());

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

- For API details, see DescribeSolution in *AWS SDK for Java 2.x API Reference*.

List campaigns

The following code example shows how to list campaigns in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

       try{
           ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
               .maxResults(10)
               .solutionArn(solutionArn)
               .build();

           ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
           List<CampaignSummary> campaigns = response.campaigns();
           for (CampaignSummary campaign: campaigns) {
               System.out.println("Campaign name is : "+campaign.name());
               System.out.println("Campaign ARN is : "+campaign.campaignArn());
           }

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

- For API details, see ListCampaigns in *AWS SDK for Java 2.x API Reference*.

List dataset groups

The following code example shows how to list dataset groups in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listDSGroups( PersonalizeClient personalizeClient ) {

       try {
           ListDatasetGroupsRequest groupsRequest = ListDatasetGroupsRequest.builder()
               .maxResults(15)
               .build();

           ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
           List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
           for (DatasetGroupSummary group: groups) {
               System.out.println("The DataSet name is : "+group.name());
               System.out.println("The DataSet ARN is : "+group.datasetGroupArn());
           }
```

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

- For API details, see ListDatasetGroups in *AWS SDK for Java 2.x API Reference*.

List recipes

The following code example shows how to list recipes in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllRecipes(PersonalizeClient personalizeClient) {

        try {
            ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
                .maxResults(15)
                .build();

            ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
            List<RecipeSummary> recipes = response.recipes();
            for (RecipeSummary recipe: recipes) {
                System.out.println("The recipe ARN is: "+recipe.recipeArn());
                System.out.println("The recipe name is: "+recipe.name());
            }

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

- For API details, see ListRecipes in *AWS SDK for Java 2.x API Reference*.

List solutions

The following code example shows how to list solutions in Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

        try {
            ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
                .maxResults(10)
                .datasetGroupArn(datasetGroupArn)
```

```
                .build() ;

            ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
            List<SolutionSummary> solutions = response.solutions();
            for (SolutionSummary solution: solutions) {
                System.out.println("The solution ARN is: "+solution.solutionArn());
                System.out.println("The solution name is: "+solution.name());
            }

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

- For API details, see ListSolutions in *AWS SDK for Java 2.x API Reference*.

Update a campaign

The following code example shows how to update a campaign Amazon Personalize.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String updateCampaign(PersonalizeClient personalizeClient,
                                        String campaignArn,
                                        String solutionVersionArn,
                                        Integer minProvisionedTPS) {

        try {
            // build the updateCampaignRequest
            UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .solutionVersionArn(solutionVersionArn)
                    .minProvisionedTPS(minProvisionedTPS)
                    .build();

            // update the campaign
            personalizeClient.updateCampaign(updateCampaignRequest);

            DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
            Campaign updatedCampaign = campaignResponse.campaign();

            System.out.println("The Campaign status is " + updatedCampaign.status());
            return updatedCampaign.status();

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

- For API details, see UpdateCampaign in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Events.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Import real-time interaction event data

The following code example shows how to import real-time interaction event data into Amazon Personalize Events.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String item1Id,
                           String item1PropertyName,
                           String item1PropertyValue,
                           String item2Id,
                           String item2PropertyName,
                           String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
                .itemId(item1Id)
                .properties(String.format("{\"%1$s\": \"%2$s\"}",
                        item1PropertyName, item1PropertyValue))
                .build();

        items.add(item1);

        Item item2 = Item.builder()
                .itemId(item2Id)
                .properties(String.format("{\"%1$s\": \"%2$s\"}",
                        item2PropertyName, item2PropertyValue))
                .build();
```

```
            items.add(item2);

            PutItemsRequest putItemsRequest = PutItemsRequest.builder()
                    .datasetArn(datasetArn)
                    .items(items)
                    .build();

            responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
            System.out.println("Response code: " + responseCode);
            return responseCode;

        } catch (PersonalizeEventsException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return responseCode;
    }
```

- For API details, see PutEvents in *AWS SDK for Java 2.x API Reference*.

Incrementally import a user

The following code example shows how to incrementally import a user into Amazon Personalize Events Events.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
                               String datasetArn,
                               String user1Id,
                               String user1PropertyName,
                               String user1PropertyValue,
                               String user2Id,
                               String user2PropertyName,
                               String user2PropertyValue) {

        int responseCode = 0;
        ArrayList<User> users = new ArrayList<>();

        try {
            User user1 = User.builder()
                    .userId(user1Id)
                    .properties(String.format("{\"%1$s\": \"%2$s\"}",
                            user1PropertyName, user1PropertyValue))
                    .build();

            users.add(user1);

            User user2 = User.builder()
                    .userId(user2Id)
                    .properties(String.format("{\"%1$s\": \"%2$s\"}",
                            user2PropertyName, user2PropertyValue))
                    .build();

            users.add(user2);
```

```
            PutUsersRequest putUsersRequest = PutUsersRequest.builder()
                    .datasetArn(datasetArn)
                    .users(users)
                    .build();

            responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
            System.out.println("Response code: " + responseCode);
            return responseCode;

        } catch (PersonalizeEventsException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return responseCode;
    }
```

- For API details, see PutUsers in *AWS SDK for Java 2.x API Reference*.

Amazon Personalize Runtime examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Runtime.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Get recommendations (custom dataset group)

The following code example shows how to get Amazon Personalize Runtime Runtime ranked recommendations.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                                    String campaignArn,
                                                    String userId,
                                                    ArrayList<String> items) {

        try {
            GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
                    .campaignArn(campaignArn)
                    .userId(userId)
                    .inputList(items)
```

```
                    .build();

            GetPersonalizedRankingResponse recommendationsResponse =

personalizeRuntimeClient.getPersonalizedRanking(rankingRecommendationsRequest);
            List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
            int rank = 1;
            for (PredictedItem item : rankedItems) {
                System.out.println("Item ranked at position " + rank + " details");
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
                System.out.println("--------------------------------------------");
                rank++;
            }
            return rankedItems;
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- For API details, see GetPersonalizedRanking in *AWS SDK for Java 2.x API Reference*.

Get recommendations from a recommender (domain dataset group)

The following code example shows how to get Amazon Personalize Runtime Runtime recommendations.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get a list of recommended items.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId){

        try {
            GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                .campaignArn(campaignArn)
                .numResults(20)
                .userId(userId)
                .build();

            GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();
            for (PredictedItem item: items) {
                System.out.println("Item Id is : "+item.itemId());
                System.out.println("Item score is : "+item.score());
            }

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

Get a list of recommended items from a recommender created in a domain dataset group.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn, String userId){

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .recommenderArn(recommenderArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();

           for (PredictedItem item: items) {
               System.out.println("Item Id is : "+item.itemId());
               System.out.println("Item score is : "+item.score());
           }
       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Use a filter when requesting recommendations.

```
   public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                      String campaignArn,
                                      String userId,
                                      String filterArn,
                                      String parameter1Name,
                                      String parameter1Value1,
                                      String parameter1Value2,
                                      String parameter2Name,
                                      String parameter2Value){

       try {

           Map<String, String> filterValues = new HashMap<>();

           filterValues.put(parameter1Name, String.format("\"%1$s\",\"%2$s\"",
                   parameter1Value1, parameter1Value2));
           filterValues.put(parameter2Name, String.format("\"%1$s\"",
                   parameter2Value));

           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .campaignArn(campaignArn)
                   .numResults(20)
                   .userId(userId)
                   .filterArn(filterArn)
                   .filterValues(filterValues)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();

           for (PredictedItem item: items) {
```

```
                System.out.println("Item Id is : "+item.itemId());
                System.out.println("Item score is : "+item.score());
            }
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetRecommendations in *AWS SDK for Java 2.x API Reference*.

Amazon Pinpoint examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a campaign

The following code example shows how to create a campaign.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a campaign.

```
   public static void createPinCampaign(PinpointClient pinpoint, String appId, String
segmentId) {
       CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
       System.out.println("Campaign " + result.name() + " created.");
       System.out.println(result.description());
   }

   public static CampaignResponse createCampaign(PinpointClient client, String appID,
String segmentID) {

       try {
           Schedule schedule = Schedule.builder()
               .startTime("IMMEDIATE")
               .build();

           Message defaultMessage = Message.builder()
               .action(Action.OPEN_APP)
               .body("My message body.")
```

```
                .title("My message title.")
                .build();

            MessageConfiguration messageConfiguration = MessageConfiguration.builder()
                .defaultMessage(defaultMessage)
                .build();

            WriteCampaignRequest request = WriteCampaignRequest.builder()
                .description("My description")
                .schedule(schedule)
                .name("MyCampaign")
                .segmentId(segmentID)
                .messageConfiguration(messageConfiguration)
                .build();

            CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
                            .applicationId(appID)
                            .writeCampaignRequest(request).build()
            );

            System.out.println("Campaign ID: " + result.campaignResponse().id());
            return result.campaignResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- For API details, see CreateCampaign in *AWS SDK for Java 2.x API Reference*.

Create a segment

The following code example shows how to create a segment.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static SegmentResponse createSegment(PinpointClient client, String appId) {

        try {
            Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
            segmentAttributes.put("Team", AttributeDimension.builder()
                .attributeType(AttributeType.INCLUSIVE)
                .values("Lakers")
                .build());

            RecencyDimension recencyDimension = RecencyDimension.builder()
                .duration("DAY_30")
                .recencyType("ACTIVE")
                .build();

            SegmentBehaviors segmentBehaviors = SegmentBehaviors.builder()
                .recency(recencyDimension)
                .build();
```

```
            SegmentDemographics segmentDemographics = SegmentDemographics
                .builder()
                .build();

            SegmentLocation segmentLocation = SegmentLocation
                .builder()
                .build();

            SegmentDimensions dimensions = SegmentDimensions
                .builder()
                .attributes(segmentAttributes)
                .behavior(segmentBehaviors)
                .demographic(segmentDemographics)
                .location(segmentLocation)
                .build();

            WriteSegmentRequest writeSegmentRequest = WriteSegmentRequest.builder()
                .name("MySegment")
                .dimensions(dimensions)
                .build();

            CreateSegmentRequest createSegmentRequest = CreateSegmentRequest.builder()
                .applicationId(appId)
                .writeSegmentRequest(writeSegmentRequest)
                .build();

            CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
            System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
            System.out.println("Done");
            return createSegmentResult.segmentResponse();

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

- For API details, see CreateSegment in *AWS SDK for Java 2.x API Reference*.

Create an application

The following code example shows how to create an appliation.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createApplication(PinpointClient pinpoint, String appName) {

        try {
            CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
                .name(appName)
                .build();

            CreateAppRequest request = CreateAppRequest.builder()
                .createApplicationRequest(appRequest)
                .build();
```

```
            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

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

- For API details, see CreateApp in *AWS SDK for Java 2.x API Reference*.

Delete an application

The following code example shows how to delete an application.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete an application.

```
    public static void deletePinApp(PinpointClient pinpoint, String appId ) {

        try {
            DeleteAppRequest appRequest = DeleteAppRequest.builder()
                .applicationId(appId)
                .build();

            DeleteAppResponse result = pinpoint.deleteApp(appRequest);
            String appName = result.applicationResponse().name();
            System.out.println("Application " + appName + " has been deleted.");

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

- For API details, see DeleteApp in *AWS SDK for Java 2.x API Reference*.

Delete an endpoint

The following code example shows how to delete an endpoint.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete an endpoint.

```
    public static void deletePinEncpoint(PinpointClient pinpoint, String appId, String
endpointId ) {
```

```
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpointId)
                .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id  " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

- For API details, see DeleteEndpoint in *AWS SDK for Java 2.x API Reference*.

Export an endpoint

The following code example shows how to export an endpoint.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Export an endpoint.

```
    public static void exportAllEndpoints(PinpointClient pinpoint,
                                          S3Client s3Client,
                                          String applicationId,
                                          String s3BucketName,
                                          String path,
                                          String iamExportRoleArn) {

        try {
            List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
 s3BucketName, iamExportRoleArn, applicationId);
            List<String> endpointFileKeys = objectKeys.stream().filter(o ->
 o.endsWith(".gz")).collect(Collectors.toList());
            downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

        } catch ( PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
 s3Client, String s3BucketName, String iamExportRoleArn, String applicationId) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
        String endpointsKeyPrefix = "exports/" + applicationId + "_" +
 dateFormat.format(new Date());
        String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix + "/";
        List<String> objectKeys = new ArrayList<>();
        String key;
```

```
        try {
            // Defines the export job that Amazon Pinpoint runs.
            ExportJobRequest jobRequest = ExportJobRequest.builder()
                .roleArn(iamExportRoleArn)
                .s3UrlPrefix(s3UrlPrefix)
                .build();

            CreateExportJobRequest exportJobRequest = CreateExportJobRequest.builder()
                .applicationId(applicationId)
                .exportJobRequest(jobRequest)
                .build();

            System.out.format("Exporting endpoints from Amazon Pinpoint application %s
to Amazon S3 " +
                    "bucket %s . . .\n", applicationId, s3BucketName);

            CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
            String jobId = exportResult.exportJobResponse().id();
            System.out.println(jobId);
            printExportJobStatus(pinpoint, applicationId, jobId);

            ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
                .bucket(s3BucketName)
                .prefix(endpointsKeyPrefix)
                .build();

            // Create a list of object keys.
            ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
            List<S3Object> objects = v2Response.contents();
            for (S3Object object: objects) {
                key = object.key();
                objectKeys.add(key);
            }

            return objectKeys;

        } catch ( PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
                                             String applicationId,
                                             String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
                .jobId(jobId)
                .applicationId(applicationId)
                .build();

            do {
                getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
                status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
                System.out.format("Export job %s . . .\n", status);
                TimeUnit.SECONDS.sleep(3);

            } while (!status.equals("COMPLETED") && !status.equals("FAILED"));
```

```
            if (status.equals("COMPLETED")) {
                System.out.println("Finished exporting endpoints.");
            } else {
                System.err.println("Failed to export endpoints.");
                System.exit(1);
            }

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

    // Download files from an Amazon S3 bucket and write them to the path location.
    public static void downloadFromS3(S3Client s3Client, String path, String
 s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(s3BucketName)
                    .key(key)
                    .build();

                ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
                String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
 Date());
                newPath = path + fileSuffix+".gz";
                File myFile = new File(newPath);
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");

        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CreateExportJob in *AWS SDK for Java 2.x API Reference*.

Get endpoints

The following code example shows how to get endpoints.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
 String endpoint ) {
```

```
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpoint)
                .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.
            Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create();

            String endpointJson = gson.toJson(endResponse);
            System.out.println(endpointJson);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

- For API details, see GetEndpoint in *AWS SDK for Java 2.x API Reference*.

Import a segment

The following code example shows how to import a segment.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Import a segment.

```
    public static ImportJobResponse createImportSegment(PinpointClient client,
                                                        String appId,
                                                        String bucket,
                                                        String key,
                                                        String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                .defineSegment(true)
                .registerEndpoints(true)
                .roleArn(roleArn)
                .format(Format.JSON)
                .s3Url("s3://" + bucket + "/" + key)
                .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                .importJobRequest(importRequest)
                .applicationId(appId)
                .build();

            CreateImportJobResponse jobResponse = client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();
```

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

- For API details, see CreateImportJob in *AWS SDK for Java 2.x API Reference*.

List endpoints

The following code example shows how to list endpoints.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllEndpoints(PinpointClient pinpoint,
                                        String applicationId,
                                       String userId) {

        try {
            GetUserEndpointsRequest endpointsRequest =
GetUserEndpointsRequest.builder()
                .userId(userId)
                .applicationId(applicationId)
                .build();

            GetUserEndpointsResponse response =
pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints = response.endpointsResponse().item();

            // Display the results.
            for (EndpointResponse endpoint: endpoints) {
                System.out.println("The channel type is: "+endpoint.channelType());
                System.out.println("The address is  "+endpoint.address());
            }

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

- For API details, see GetUserEndpoints in *AWS SDK for Java 2.x API Reference*.

List segments

The following code example shows how to list segments.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

List segments.

```
    public static void listSegs( PinpointClient pinpoint, String appId) {

        try {
            GetSegmentsRequest request = GetSegmentsRequest.builder()
                .applicationId(appId)
                .build();

            GetSegmentsResponse response = pinpoint.getSegments(request);
            List<SegmentResponse> segments = response.segmentsResponse().item();
            for(SegmentResponse segment: segments) {
                System.out.println("Segement " + segment.id() + " " + segment.name() +
 " " + segment.lastModifiedDate());
            }

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

- For API details, see GetSegments in *AWS SDK for Java 2.x API Reference*.

Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Send an email message.

```
    public static void sendEmail(PinpointClient pinpoint,
                                 String subject,
                                 String appId,
                                 String senderAddress,
                                 String toAddress) {

        try {
            Map<String,AddressConfiguration> addressMap = new HashMap<>();
            AddressConfiguration configuration = AddressConfiguration.builder()
                .channelType(ChannelType.EMAIL)
                .build();

            addressMap.put(toAddress, configuration);
            SimpleEmailPart emailPart = SimpleEmailPart.builder()
                .data(htmlBody)
                .charset(charset)
                .build() ;

            SimpleEmailPart subjectPart = SimpleEmailPart.builder()
                .data(subject)
                .charset(charset)
                .build() ;

            SimpleEmail simpleEmail = SimpleEmail.builder()
                .htmlPart(emailPart)
                .subject(subjectPart)
                .build();
```

```
            EmailMessage emailMessage = EmailMessage.builder()
                .body(htmlBody)
                .fromAddress(senderAddress)
                .simpleEmail(simpleEmail)
                .build();

            DirectMessageConfiguration directMessageConfiguration =
DirectMessageConfiguration.builder()
                .emailMessage(emailMessage)
                .build();

            MessageRequest messageRequest = MessageRequest.builder()
                .addresses(addressMap)
                .messageConfiguration(directMessageConfiguration)
                .build();

            SendMessagesRequest messagesRequest = SendMessagesRequest.builder()
                .applicationId(appId)
                .messageRequest(messageRequest)
                .build();

            pinpoint.sendMessages(messagesRequest);

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

Send an SMS message.

```
    public static void sendSMSMessage(PinpointClient pinpoint, String message, String
appId, String originationNumber, String destinationNumber) {

        try {
            Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
            AddressConfiguration addConfig = AddressConfiguration.builder()
                .channelType(ChannelType.SMS)
                .build();

            addressMap.put(destinationNumber, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                .body(message)
                .messageType(messageType)
                .originationNumber(originationNumber)
                .senderId(senderId)
                .keyword(registeredKeyword)
                .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
                .smsMessage(smsMessage)
                .build();

            MessageRequest msgReq = MessageRequest.builder()
                .addresses(addressMap)
                .messageConfiguration(direct)
                .build();

            // create a  SendMessagesRequest object
            SendMessagesRequest request = SendMessagesRequest.builder()
                .applicationId(appId)
                .messageRequest(msgReq)
```

```
                .build();

            SendMessagesResponse response= pinpoint.sendMessages(request);
            MessageResponse msg1 = response.messageResponse();
            Map map1 = msg1.result();

            //Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

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

Send batch SMS messages.

```
   public static void sendSMSMessage(PinpointClient pinpoint, String message, String
appId, String originationNumber, String destinationNumber, String destinationNumber1)
{
        try {
            Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
            AddressConfiguration addConfig = AddressConfiguration.builder()
                .channelType(ChannelType.SMS)
                .build();

            // Add an entry to the Map object for each number to whom you want to send
a message.
            addressMap.put(destinationNumber, addConfig);
            addressMap.put(destinationNumber1, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                .body(message)
                .messageType(messageType)
                .originationNumber(originationNumber)
                .senderId(senderId)
                .keyword(registeredKeyword)
                .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
                .smsMessage(smsMessage)
                .build();

            MessageRequest msgReq = MessageRequest.builder()
                .addresses(addressMap)
                .messageConfiguration(direct)
                .build();

            // Create a SendMessagesRequest object.
            SendMessagesRequest request = SendMessagesRequest.builder()
                .applicationId(appId)
                .messageRequest(msgReq)
                .build();

            SendMessagesResponse response= pinpoint.sendMessages(request);
            MessageResponse msg1 = response.messageResponse();
            Map map1 = msg1.result();

            // Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

- For API details, see SendMessages in *AWS SDK for Java 2.x API Reference*.

Update an endpoint

The following code example shows how to update an endpoint.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static EndpointResponse createEndpoint(PinpointClient client, String appId)
{
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        try {
            EndpointRequest endpointRequest = createEndpointRequestData();
            UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpointId)
                .endpointRequest(endpointRequest)
                .build();

            UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
            System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

            GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
                .applicationId(appId)
                .endpointId(endpointId)
                .build();

            GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
            System.out.println(getEndpointResponse.endpointResponse().address());
            System.out.println(getEndpointResponse.endpointResponse().channelType());
            System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
            System.out.println(getEndpointResponse.endpointResponse().requestId());
            System.out.println(getEndpointResponse.endpointResponse().user());

            return getEndpointResponse.endpointResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static EndpointRequest createEndpointRequestData() {
```

```
        try {
            List<String> favoriteTeams = new ArrayList<>();
            favoriteTeams.add("Lakers");
            favoriteTeams.add("Warriors");
            HashMap<String, List<String>> customAttributes = new HashMap<>();
            customAttributes.put("team", favoriteTeams);

            EndpointDemographic demographic = EndpointDemographic.builder()
                .appVersion("1.0")
                .make("apple")
                .model("iPhone")
                .modelVersion("7")
                .platform("ios")
                .platformVersion("10.1.1")
                .timezone("America/Los_Angeles")
                .build();

            EndpointLocation location = EndpointLocation.builder()
                .city("Los Angeles")
                .country("US")
                .latitude(34.0)
                .longitude(-118.2)
                .postalCode("90068")
                .region("CA")
                .build();

            Map<String,Double> metrics = new HashMap<>();
            metrics.put("health", 100.00);
            metrics.put("luck", 75.00);

            EndpointUser user = EndpointUser.builder()
                .userId(UUID.randomUUID().toString())
                .build();

            DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
"Z" to indicate UTC, no timezone offset
            String nowAsISO = df.format(new Date());

            return EndpointRequest.builder()
                .address(UUID.randomUUID().toString())
                .attributes(customAttributes)
                .channelType("APNS")
                .demographic(demographic)
                .effectiveDate(nowAsISO)
                .location(location)
                .metrics(metrics)
                .optOut("NONE")
                .requestId(UUID.randomUUID().toString())
                .user(user)
                .build();

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

- For API details, see UpdateEndpoint in *AWS SDK for Java 2.x API Reference*.

Update channels

The following code example shows how to update channels.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   private static SMSChannelResponse getSMSChannel(PinpointClient client, String
appId) {

       try {
           GetSmsChannelRequest request = GetSmsChannelRequest.builder()
               .applicationId(appId)
               .build();

           SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
           System.out.println("Channel state is " + response.enabled());
           return response;

       } catch ( PinpointException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
   }

   private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
       boolean enabled = !getResponse.enabled();

       try {
           SMSChannelRequest request = SMSChannelRequest.builder()
               .enabled(enabled)
               .build();

           UpdateSmsChannelRequest updateRequest = UpdateSmsChannelRequest.builder()
               .smsChannelRequest(request)
               .applicationId(appId)
               .build();

           UpdateSmsChannelResponse result = client.updateSmsChannel(updateRequest);
           System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

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

- For API details, see GetSmsChannel in *AWS SDK for Java 2.x API Reference*.

Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint SMS and Voice API.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Send a voice message with Amazon Pinpoint SMS and Voice API

The following code example shows how to send a voice message with Amazon Pinpoint SMS and Voice API.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber, String destinationNumber ) {

       try {
           SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
               .languageCode(languageCode)
               .text(ssmlMessage)
               .voiceId(voiceName)
               .build();

           VoiceMessageContent content = VoiceMessageContent.builder()
               .ssmlMessage(ssmlMessageType)
               .build();

           SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
               .destinationPhoneNumber(destinationNumber)
               .originationPhoneNumber(originationNumber)
               .content(content)
               .build();

           client.sendVoiceMessage(voiceMessageRequest);
           System.out.println("The message was sent successfully.");

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

- For API details, see SendVoiceMessage in *AWS SDK for Java 2.x API Reference*.

Amazon RDS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon RDS.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a DB instance

The following code example shows how to create an Amazon RDS DB instance and wait for it to become available.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createSnapshot(RdsClient rdsClient, String dbInstanceIdentifier,
String dbSnapshotIdentifier) {

       try {
           CreateDbSnapshotRequest snapshotRequest = CreateDbSnapshotRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .dbSnapshotIdentifier(dbSnapshotIdentifier)
               .build();

           CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
           System.out.print("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

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

- For API details, see CreateDBInstance in *AWS SDK for Java 2.x API Reference*.

Create a DB parameter group

The following code example shows how to create an Amazon RDS DB parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void createDBParameterGroup(RdsClient rdsClient, String dbGroupName,
String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
               .dbParameterGroupName(dbGroupName)
               .dbParameterGroupFamily(dbParameterGroupFamily)
               .description("Created by using the AWS SDK for Java")
               .build();

           CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
           System.out.println("The group name is "+
response.dbParameterGroup().dbParameterGroupName());

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

- For API details, see CreateDBParameterGroup in *AWS SDK for Java 2.x API Reference*.

Create a snapshot of a DB instance

The following code example shows how to create a snapshot of an Amazon RDS DB instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Create an Amazon RDS snapshot.
   public static void createSnapshot(RdsClient rdsClient, String dbInstanceIdentifier,
String dbSnapshotIdentifier) {
       try {
           CreateDbSnapshotRequest snapshotRequest = CreateDbSnapshotRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .dbSnapshotIdentifier(dbSnapshotIdentifier)
               .build();

           CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
           System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

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

- For API details, see CreateDBSnapshot in *AWS SDK for Java 2.x API Reference*.

Create an authentication token

The following code example shows how to create an authentication token for IAM authentication.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use the RdsUtilities class to generate an authentication token.

```
public class GenerateRDSAuthToken {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage:\n" +
            "    <dbInstanceIdentifier> <masterUsername>\n\n" +
            "Where:\n" +
            "    dbInstanceIdentifier - The database instance identifier. \n" +
            "    masterUsername - The master user name. \n";

            if (args.length != 2) {
                System.out.println(usage);
                System.exit(1);
            }

            String dbInstanceIdentifier = args[0];
            String masterUsername = args[1];
            Region region = Region.US_WEST_2;
            RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

            String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
            System.out.println("The token response is "+token);
    }

    public static String getAuthToken(RdsClient rdsClient, String dbInstanceIdentifier,
 String masterUsername ) {

        RdsUtilities utilities = rdsClient.utilities();
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .username(masterUsername)
                .port(3306)
                .hostname(dbInstanceIdentifier)
                .build();

                return utilities.generateAuthenticationToken(tokenRequest);

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

- For API details, see GenerateRDSAuthToken in *AWS SDK for Java 2.x API Reference*.

Delete a DB instance

The following code example shows how to delete an Amazon RDS DB instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see DeleteDBInstance in *AWS SDK for Java 2.x API Reference*.

Delete a DB parameter group

The following code example shows how to delete an Amazon RDS DB parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Delete the parameter group after database has been deleted.
   // An exception is thrown if you attempt to delete the para group while database
exists.
   public static void deleteParaGroup( RdsClient rdsClient, String dbGroupName, String
dbARN) throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN ;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
```

```
                isDataDel = false ;
                didFind = false;
                int index = 1;
                for (DBInstance instance: instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true ;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the database
ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index ++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName +" was deleted.");

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

- For API details, see DeleteDBParameterGroup in *AWS SDK for Java 2.x API Reference*.

Describe DB instances

The following code example shows how to describe Amazon RDS DB instances.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void describeInstances(RdsClient rdsClient) {

        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance: instanceList) {
                System.out.println("Instance ARN is: "+instance.dbInstanceArn());
                System.out.println("The Engine is " +instance.engine());
                System.out.println("Connection endpoint is"
+instance.endpoint().address());
            }

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

```
        }
    }
```

- For API details, see DescribeDBInstances in *AWS SDK for Java 2.x API Reference*.

Describe DB parameter groups

The following code example shows how to describe Amazon RDS DB parameter groups.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
       try {
           DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
               .dbParameterGroupName(dbGroupName)
               .maxRecords(20)
               .build();

           DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
           List<DBParameterGroup> groups = response.dbParameterGroups();
           for (DBParameterGroup group: groups) {
               System.out.println("The group name is "+group.dbParameterGroupName());
               System.out.println("The group description is "+group.description());
           }

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

- For API details, see DescribeDBParameterGroups in *AWS SDK for Java 2.x API Reference*.

Describe database engine versions

The following code example shows how to describe Amazon RDS database engine versions.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeDBEngines( RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
               .defaultOnly(true)
               .engine("mysql")
```

```
                .maxRecords(20)
                .build();

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

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

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

- For API details, see DescribeDBEngineVersions in *AWS SDK for Java 2.x API Reference*.

Describe options for DB instances

The following code example shows how to describe options for Amazon RDS DB instances.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

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

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

- For API details, see DescribeOrderableDBInstanceOptions in *AWS SDK for Java 2.x API Reference*.

Describe parameters in a DB parameter group

The following code example shows how to describe parameters in an Amazon RDS DB parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
  // Retrieve parameters in the group.
  public static void describeDbParameters(RdsClient rdsClient, String dbGroupName, int
flag) {
      try {
          DescribeDbParametersRequest dbParameterGroupsRequest;
          if (flag == 0) {
              dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                  .dbParameterGroupName(dbGroupName)
                  .build();
          } else {
              dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                  .dbParameterGroupName(dbGroupName)
                  .source("user")
                  .build();
          }

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

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

- For API details, see DescribeDBParameters in *AWS SDK for Java 2.x API Reference*.

Modify a DB instance

The following code example shows how to modify an Amazon RDS DB instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void updateIntance(RdsClient rdsClient, String dbInstanceIdentifier,
String masterUserPassword) {

       try {
           // For a demo - modify the DB instance by modifying the master password.
           ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .publiclyAccessible(true)
               .masterUserPassword(masterUserPassword)
               .build();

           ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);
           System.out.print("The ARN of the modified database is: "
+instanceResponse.dbInstance().dbInstanceArn());

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

- For API details, see ModifyDBInstance in *AWS SDK for Java 2.x API Reference*.

Reboot a DB instance

The following code example shows how to reboot an Amazon RDS DB instance.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier ) {

       try {
           RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
              .dbInstanceIdentifier(dbInstanceIdentifier)
              .build();

           RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
           System.out.print("The database "+
instanceResponse.dbInstance().dbInstanceArn() +" was rebooted");

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

- For API details, see RebootDBInstance in *AWS SDK for Java 2.x API Reference*.

Retrieve attributes

The following code example shows how to retrieve attributes that belong to an Amazon RDS account.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getAccountAttributes(RdsClient rdsClient) {

       try {
           DescribeAccountAttributesResponse response =
rdsClient.describeAccountAttributes();
           List<AccountQuota> quotasList = response.accountQuotas();
           for (AccountQuota quotas: quotasList) {
               System.out.println("Name is: "+quotas.accountQuotaName());
               System.out.println("Max value is " +quotas.max());
           }

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

- For API details, see DescribeAccountAttributes in *AWS SDK for Java 2.x API Reference*.

Update parameters in a DB parameter group

The following code example shows how to update parameters in an Amazon RDS DB parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   // Modify auto_increment_offset and auto_increment_increment parameters.
   public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
       try {
           Parameter parameter1 = Parameter.builder()
               .parameterName("auto_increment_offset")
               .applyMethod("immediate")
               .parameterValue("5")
               .build();

           List<Parameter> paraList = new ArrayList<>();
           paraList.add(parameter1);
           ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
               .dbParameterGroupName(dbGroupName)
               .parameters(paraList)
               .build();
```

```
            ModifyDbParameterGroupResponse response =
 rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group "+ response.dbParameterGroupName()
 +" was successfully modified");

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

- For API details, see ModifyDBParameterGroup in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with DB instances

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.
- Delete the instance and parameter group.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run multiple operations.

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses the
 parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection endpoint
 value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the  RDS DB instance.
```

```
 * 14. Deletes the parameter group.
 */
public class RDSScenario {

    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
 <masterUsername> <masterUserPassword> <dbSnapshotIdentifier>\n\n" +
            "Where:\n" +
            "    dbGroupName - The database group name. \n"+
            "    dbParameterGroupFamily - The database parameter group name (for
 example, mysql8.0).\n"+
            "    dbInstanceIdentifier - The database instance identifier \n"+
            "    dbName - The database name. \n"+
            "    masterUsername - The master user name. \n"+
            "    masterUserPassword - The password that corresponds to the master user
 name. \n"+
            "    dbSnapshotIdentifier - The snapshot identifier. \n" ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String masterUsername = args[4];
        String masterUserPassword = args[5];
        String dbSnapshotIdentifier = args[6];

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon RDS example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);

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

```
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

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

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for the
selected engine") ;
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Create an RDS database instance that contains a MySql
database and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername, masterUserPassword);
        System.out.println("The ARN of the new database is "+dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready" );
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready" );
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance" );
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

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

        rdsClient.close();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
exists.
    public static void deleteParaGroup( RdsClient rdsClient, String dbGroupName, String
dbARN) throws InterruptedException {
        try {
```

```
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN ;

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

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName +" was deleted.");

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

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

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

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

    // Waits until the snapshot instance is available.
```

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

           DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
               .dbSnapshotIdentifier(dbSnapshotIdentifier)
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .build();

           while (!snapshotReady) {
               DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
               List<DBSnapshot> snapshotList = response.dbSnapshots();
               for (DBSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
                   } else {
                       System.out.print(".");
                       Thread.sleep(sleepTime * 1000);
                   }
               }
           }

           System.out.println("The Snapshot is available!");
       } catch (RdsException | InterruptedException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   // Create an Amazon RDS snapshot.
   public static void createSnapshot(RdsClient rdsClient, String dbInstanceIdentifier,
String dbSnapshotIdentifier) {
       try {
           CreateDbSnapshotRequest snapshotRequest = CreateDbSnapshotRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .dbSnapshotIdentifier(dbSnapshotIdentifier)
               .build();

           CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
           System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

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

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

       try {
           DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
```

```
                .build();

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

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

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
                                                String dbGroupName,
                                                String dbInstanceIdentifier,
                                                String dbName,
                                                String masterUsername,
                                                String masterUserPassword) {

        try {
            CreateDbInstanceRequest instanceRequest = CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .dbParameterGroupName(dbGroupName)
                .engine("mysql")
                .dbInstanceClass("db.m4.large")
                .engineVersion("8.0")
                .storageType("standard")
                .masterUsername(masterUsername)
                .masterUserPassword(masterUserPassword)
                .build();

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

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

        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
```

```
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest.builder()
                .engine("mysql")
                .build();

            DescribeOrderableDbInstanceOptionsResponse response =
rdsClient.describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption: orderableDBInstances) {
                System.out.println("The engine version is "
+dbInstanceOption.engineVersion());
                System.out.println("The engine description is "
+dbInstanceOption.engine());
            }

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

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

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

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

    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                .parameterName("auto_increment_offset")
                .applyMethod("immediate")
                .parameterValue("5")
                .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .parameters(paraList)
                .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
```

```
            System.out.println("The parameter group "+ response.dbParameterGroupName()
+" was successfully modified");

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

  // Retrieve parameters in the group.
  public static void describeDbParameters(RdsClient rdsClient, String dbGroupName, int
flag) {
      try {
          DescribeDbParametersRequest dbParameterGroupsRequest;
          if (flag == 0) {
              dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                  .dbParameterGroupName(dbGroupName)
                  .build();
          } else {
              dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                  .dbParameterGroupName(dbGroupName)
                  .source("user")
                  .build();
          }

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

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

   public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
       try {
           DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
               .dbParameterGroupName(dbGroupName)
               .maxRecords(20)
               .build();

           DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
           List<DBParameterGroup> groups = response.dbParameterGroups();
           for (DBParameterGroup group: groups) {
```

```
                System.out.println("The group name is "+group.dbParameterGroupName());
                System.out.println("The group description is "+group.description());
            }

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

    public static void createDBParameterGroup(RdsClient rdsClient, String dbGroupName,
 String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
 CreateDbParameterGroupRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .dbParameterGroupFamily(dbParameterGroupFamily)
                .description("Created by using the AWS SDK for Java")
                .build();

            CreateDbParameterGroupResponse response =
 rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is "+
 response.dbParameterGroup().dbParameterGroupName());

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


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

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

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

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateDBInstance

- CreateDBParameterGroup
- CreateDBSnapshot
- DeleteDBInstance
- DeleteDBParameterGroup
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Amazon Redshift examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Redshift.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a cluster

The following code example shows how to create an Amazon Redshift cluster.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the cluster.

```
   public static void createCluster(RedshiftClient redshiftClient, String clusterId,
String masterUsername, String masterUserPassword ) {

       try {
           CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
               .clusterIdentifier(clusterId)
               .masterUsername(masterUsername) // set the user name here
               .masterUserPassword(masterUserPassword) // set the user password here
               .nodeType("ds2.xlarge")
               .publiclyAccessible(true)
               .numberOfNodes(2)
               .build();
```

```
        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

      } catch (RedshiftException e) {

          System.err.println(e.getMessage());
          System.exit(1);
      }
   }
```

- For API details, see CreateCluster in *AWS SDK for Java 2.x API Reference*.

Delete a cluster

The following code example shows how to delete an Amazon Redshift cluster.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Delete the cluster.

```
   public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {

       try {
           DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
               .clusterIdentifier(clusterId)
               .skipFinalClusterSnapshot(true)
               .build();

           DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
           System.out.println("The status is "+response.cluster().clusterStatus());

       } catch (RedshiftException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see DeleteCluster in *AWS SDK for Java 2.x API Reference*.

Describe your clusters

The following code example shows how to describe your Amazon Redshift clusters.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Describe the cluster.

```
   public static void describeRedshiftClusters(RedshiftClient redshiftClient) {

      try {
           DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters();
           List<Cluster> clusterList = clusterResponse.clusters();
           for (Cluster cluster: clusterList) {
               System.out.println("Cluster database name is: "+cluster.dbName());
               System.out.println("Cluster status is: "+cluster.clusterStatus());
           }

      } catch (RedshiftException e) {
          System.err.println(e.getMessage());
          System.exit(1);
      }
  }
```

- For API details, see DescribeClusters in *AWS SDK for Java 2.x API Reference*.

Modify a cluster

The following code example shows how to modify an Amazon Redshift cluster.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Modify a cluster.

```
    public static void modifyCluster(RedshiftClient redshiftClient, String clusterId)
{

       try {
           ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
               .clusterIdentifier(clusterId)
                .preferredMaintenanceWindow("wed:07:30-wed:08:00")
               .build();

           ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
           System.out.println("The modified cluster was successfully modified and
has "+ clusterResponse.cluster().preferredMaintenanceWindow() +" as the maintenance
window");

       } catch (RedshiftException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see ModifyCluster in *AWS SDK for Java 2.x API Reference*.

Amazon Rekognition examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Rekognition.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Compare faces in an image against a reference image

The following code example shows how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see Comparing faces in images.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage, String targetImage) {
       try {
           InputStream sourceStream = new FileInputStream(sourceImage);
           InputStream tarStream = new FileInputStream(targetImage);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
           SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

           // Create an Image object for the source image.
           Image souImage = Image.builder()
               .bytes(sourceBytes)
               .build();

           Image tarImage = Image.builder()
               .bytes(targetBytes)
               .build();

           CompareFacesRequest facesRequest = CompareFacesRequest.builder()
               .sourceImage(souImage)
               .targetImage(tarImage)
               .similarityThreshold(similarityThreshold)
               .build();

           // Compare the two images.
           CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
           List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
           for (CompareFacesMatch match: faceDetails){
               ComparedFace face= match.face();
               BoundingBox position = face.boundingBox();
               System.out.println("Face at " + position.left().toString()
                       + " " + position.top()
                       + " matches with " + face.confidence().toString()
                       + "% confidence.");
```

```
            }
            List<ComparedFace> uncompared = compareFacesResult.unmatchedFaces();
            System.out.println("There was " + uncompared.size() + " face(s) that did
not match");
            System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
            System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

        } catch(RekognitionException | FileNotFoundException e) {
            System.out.println("Failed to load source image " + sourceImage);
            System.exit(1);
        }
    }
```

- For API details, see CompareFaces in *AWS SDK for Java 2.x API Reference*.

Create a collection

The following code example shows how to create an Amazon Rekognition collection.

For more information, see Creating a collection.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createMyCollection(RekognitionClient rekClient,String
collectionId ) {

        try {
            CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " + collectionResponse.collectionArn());
            System.out.println("Status code: " +
collectionResponse.statusCode().toString());

        } catch(RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CreateCollection in *AWS SDK for Java 2.x API Reference*.

Delete a collection

The following code example shows how to delete an Amazon Rekognition collection.

For more information, see Deleting a collection.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteMyCollection(RekognitionClient rekClient,String
collectionId ) {

       try {
           DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
               .collectionId(collectionId)
               .build();

           DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
           System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

       } catch(RekognitionException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see DeleteCollection in *AWS SDK for Java 2.x API Reference*.

Delete faces from a collection

The following code example shows how to delete faces from an Amazon Rekognition collection.

For more information, see Deleting faces from a collection.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteFacesCollection(RekognitionClient rekClient,
                                            String collectionId,
                                            String faceId) {

       try {
           DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
               .collectionId(collectionId)
               .faceIds(faceId)
               .build();

           rekClient.deleteFaces(deleteFacesRequest);
           System.out.println("The face was deleted from the collection.");

       } catch(RekognitionException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see DeleteFaces in *AWS SDK for Java 2.x API Reference*.

Describe a collection

The following code example shows how to describe an Amazon Rekognition collection.

For more information, see Describing a collection.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeColl(RekognitionClient rekClient, String collectionName)
{

       try {
           DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
               .collectionId(collectionName)
               .build();

           DescribeCollectionResponse describeCollectionResponse =
rekClient.describeCollection(describeCollectionRequest);
           System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
           System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

       } catch(RekognitionException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see DescribeCollection in *AWS SDK for Java 2.x API Reference*.

Detect faces in an image

The following code example shows how to detect faces in an image with Amazon Rekognition.

For more information, see Detecting faces in an image.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void detectFacesinImage(RekognitionClient rekClient,String
sourceImage ) {

       try {
           InputStream sourceStream = new FileInputStream(sourceImage);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           // Create an Image object for the source image.
           Image souImage = Image.builder()
```

```
                .bytes(sourceBytes)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(souImage)
                .build();

            DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between "
                            + ageRange.low().toString() + " and " +
ageRange.high().toString()
                            + " years old.");

                System.out.println("There is a smile :
"+face.smile().value().toString());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DetectFaces in *AWS SDK for Java 2.x API Reference*.

Detect labels in an image

The following code example shows how to detect labels in an image with Amazon Rekognition.

For more information, see Detecting labels in an image.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {

        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                .image(souImage)
                .maxLabels(10)
                .build();

            DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
```

```
            for (Label label: labels) {
                System.out.println(label.name() + ": " +
label.confidence().toString());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see DetectLabels in *AWS SDK for Java 2.x API Reference*.

Detect moderation labels in an image

The following code example shows how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see Detecting inappropriate images.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void detectModLabels(RekognitionClient rekClient, String sourceImage)
{

        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                .image(souImage)
                .minConfidence(60F)
                .build();

            DetectModerationLabelsResponse moderationLabelsResponse =
rekClient.detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels = moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");

            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                    + "\n Confidence: " + label.confidence().toString() + "%"
                    + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
```

- For API details, see DetectModerationLabels in *AWS SDK for Java 2.x API Reference*.

Detect text in an image

The following code example shows how to detect text in an image with Amazon Rekognition.

For more information, see Detecting text in an image.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {

       try {
           InputStream sourceStream = new FileInputStream(sourceImage);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
           Image souImage = Image.builder()
               .bytes(sourceBytes)
               .build();

           DetectTextRequest textRequest = DetectTextRequest.builder()
               .image(souImage)
               .build();

           DetectTextResponse textResponse = rekClient.detectText(textRequest);
           List<TextDetection> textCollection = textResponse.textDetections();
           System.out.println("Detected lines and words");
           for (TextDetection text: textCollection) {
               System.out.println("Detected: " + text.detectedText());
               System.out.println("Confidence: " + text.confidence().toString());
               System.out.println("Id : " + text.id());
               System.out.println("Parent Id: " + text.parentId());
               System.out.println("Type: " + text.type());
               System.out.println();
           }

       } catch (RekognitionException | FileNotFoundException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see DetectText in *AWS SDK for Java 2.x API Reference*.

Index faces to a collection

The following code example shows how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see Adding faces to a collection.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
```

```
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                .collectionId(collectionId)
                .image(souImage)
                .maxFaces(1)
                .qualityFilter(QualityFilter.AUTO)
                .detectionAttributes(Attribute.DEFAULT)
                .build();

            IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
            System.out.println("Results for the image");
            System.out.println("\n Faces indexed:");
            List<FaceRecord> faceRecords = facesResponse.faceRecords();
            for (FaceRecord faceRecord : faceRecords) {
                System.out.println("  Face ID: " + faceRecord.face().faceId());
                System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
            }

            List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
            System.out.println("Faces not indexed:");
            for (UnindexedFace unindexedFace : unindexedFaces) {
                System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
                System.out.println("  Reasons:");
                for (Reason reason : unindexedFace.reasons()) {
                    System.out.println("Reason:  " + reason);
                }
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see IndexFaces in *AWS SDK for Java 2.x API Reference*.

List collections

The following code example shows how to list Amazon Rekognition collections.

For more information, see Listing collections.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
```

```
                .maxResults(10)
                .build();

            ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListCollections in *AWS SDK for Java 2.x API Reference*.

List faces in a collection

The following code example shows how to list faces in an Amazon Rekognition collection.

For more information, see Listing faces in a collection.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listFacesCollection(RekognitionClient rekClient, String
collectionId ) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                .collectionId(collectionId)
                .maxResults(10)
                .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face: faces) {
                System.out.println("Confidence level there is a face:
"+face.confidence());
                System.out.println("The face Id value is "+face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
         }
      }
```

- For API details, see ListFaces in *AWS SDK for Java 2.x API Reference*.

Recognize celebrities in an image

The following code example shows how to recognize celebrities in an image with Amazon Rekognition.

For more information, see Recognizing celebrities in an image.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {

       try {
           InputStream sourceStream = new FileInputStream(sourceImage);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
           Image souImage = Image.builder()
               .bytes(sourceBytes)
               .build();

           RecognizeCelebritiesRequest request = RecognizeCelebritiesRequest.builder()
               .image(souImage)
               .build();

           RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request) ;
           List<Celebrity> celebs=result.celebrityFaces();
           System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
           for (Celebrity celebrity: celebs) {
               System.out.println("Celebrity recognized: " + celebrity.name());
               System.out.println("Celebrity ID: " + celebrity.id());

               System.out.println("Further information (if available):");
               for (String url: celebrity.urls()){
                   System.out.println(url);
               }
               System.out.println();
           }
           System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

       } catch (RekognitionException | FileNotFoundException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see RecognizeCelebrities in *AWS SDK for Java 2.x API Reference*.

Search for faces in a collection

The following code example shows how to search for faces in an Amazon Rekognition collection that match another face from the collection.

For more information, see Searching for a face (face ID).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void searchFaceInCollection(RekognitionClient rekClient,String
collectionId, String sourceImage) {
```

```
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
                .image(souImage)
                .maxFaces(10)
                .faceMatchThreshold(70F)
                .collectionId(collectionId)
                .build();

            SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest) ;
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face: faceImageMatches) {
                System.out.println("The similarity level is  "+face.similarity());
                System.out.println();
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see SearchFaces in *AWS SDK for Java 2.x API Reference*.

Search for faces in a collection compared to a reference image

The following code example shows how to search for faces in an Amazon Rekognition collection compared to a reference image.

For more information, see Searching for a face (image).

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void searchFacebyId(RekognitionClient rekClient,String collectionId,
String faceId) {

        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                .collectionId(collectionId)
                .faceId(faceId)
                .faceMatchThreshold(70F)
                .maxFaces(2)
                .build();

            SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest) ;
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
```

```
            for (FaceMatch face: faceImageMatches) {
                System.out.println("The similarity level is  "+face.similarity());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see SearchFacesByImage in *AWS SDK for Java 2.x API Reference*.

Scenarios

Detect information in videos

The following code example shows how to:

- Start Amazon Rekognition jobs to detect elements like people, objects, and text in videos.
- Check job status until jobs finish.
- Output the list of elements detected by each job.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Get celebrity results from a video located in an Amazon S3 bucket.

```
    public static void StartCelebrityDetection(RekognitionClient rekClient,
                                               NotificationChannel channel,
                                               String bucket,
                                               String video){
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidOb = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                .jobTag("Celebrities")
                .notificationChannel(channel)
                .video(vidOb)
                .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient.startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch(RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    public static void GetCelebrityDetectionResults(RekognitionClient rekClient) {

        try {
            String paginationToken=null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy=0 ;

            do{
                if (recognitionResponse !=null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                    .jobId(startJobId)
                    .nextToken(paginationToken)
                    .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                    .maxResults(10)
                    .build();

                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData=recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs= recognitionResponse.celebrities();
                for (CelebrityRecognition celeb: celebs) {
                    long seconds=celeb.timestamp()/1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details=celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

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

Detect labels in a video by a label detection operation.

```
   public static void startLabels(RekognitionClient rekClient,
                                  NotificationChannel channel,
                                  String bucket,
                                  String video) {
       try {
           S3Object s3Obj = S3Object.builder()
               .bucket(bucket)
               .name(video)
               .build();

           Video vidOb = Video.builder()
               .s3Object(s3Obj)
               .build();

           StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
               .jobTag("DetectingLabels")
               .notificationChannel(channel)
               .video(vidOb)
               .minConfidence(50F)
               .build();

           StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
           startJobId = labelDetectionResponse.jobId();

           boolean ans = true;
           String status = "";
           int yy = 0;
           while (ans) {

               GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                   .jobId(startJobId)
                   .maxResults(10)
                   .build();

               GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
               status = result.jobStatusAsString();

               if (status.compareTo("SUCCEEDED") == 0)
                   ans = false;
               else
                   System.out.println(yy +" status is: "+status);

               Thread.sleep(1000);
               yy++;
           }

           System.out.println(startJobId +" status is: "+status);

       } catch(RekognitionException | InterruptedException e) {
           e.getMessage();
           System.exit(1);
       }
   }

   public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs, String
queueUrl) {
```

```
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message: messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId)==0) {
                        System.out.println("Job id: " + operationJobId );
                        System.out.println("Status : " + operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            GetResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    }

                    else{
                        System.out.println("Job received was not job " +  startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch(RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void GetResultsLabels(RekognitionClient rekClient) {

        int maxResults=10;
        String paginationToken=null;
        GetLabelDetectionResponse labelDetectionResult=null;

        try {
```

```
            do {
                if (labelDetectionResult !=null)
                    paginationToken = labelDetectionResult.nextToken();


                GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                    .jobId(startJobId)
                    .sortBy(LabelDetectionSortBy.TIMESTAMP)
                    .maxResults(maxResults)
                    .nextToken(paginationToken)
                    .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels= labelDetectionResult.labels();
                for (LabelDetection detectedLabel: detectedLabels) {
                    long seconds=detectedLabel.timestamp();
                    Label label=detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() + ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult !=null && labelDetectionResult.nextToken() !=
null);

        } catch(RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
```

Detect faces in a video stored in an Amazon S3 bucket.

```
   public static void startLabels(RekognitionClient rekClient,
                                  NotificationChannel channel,
                                  String bucket,
                                  String video) {
       try {
           S3Object s3Obj = S3Object.builder()
               .bucket(bucket)
               .name(video)
               .build();

           Video vidOb = Video.builder()
               .s3Object(s3Obj)
               .build();

           StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
               .jobTag("DetectingLabels")
               .notificationChannel(channel)
               .video(vidOb)
               .minConfidence(50F)
               .build();

           StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
           startJobId = labelDetectionResponse.jobId();

           boolean ans = true;
           String status = "";
           int yy = 0;
           while (ans) {

               GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                   .jobId(startJobId)
                   .maxResults(10)
                   .build();

               GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
               status = result.jobStatusAsString();

               if (status.compareTo("SUCCEEDED") == 0)
                   ans = false;
               else
                   System.out.println(yy +" status is: "+status);

               Thread.sleep(1000);
               yy++;
           }

           System.out.println(startJobId +" status is: "+status);

       } catch(RekognitionException | InterruptedException e) {
           e.getMessage();
           System.exit(1);
       }
   }

   public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs, String
queueUrl) {

       List<Message> messages;
       ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
```

```
            .queueUrl(queueUrl)
            .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message: messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId)==0) {
                        System.out.println("Job id: " + operationJobId );
                        System.out.println("Status : " + operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            GetResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    }

                    else{
                        System.out.println("Job received was not job " +  startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch(RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void GetResultsLabels(RekognitionClient rekClient) {

        int maxResults=10;
        String paginationToken=null;
        GetLabelDetectionResponse labelDetectionResult=null;

        try {
            do {
                if (labelDetectionResult !=null)
```

```
                    paginationToken = labelDetectionResult.nextToken();


                GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                    .jobId(startJobId)
                    .sortBy(LabelDetectionSortBy.TIMESTAMP)
                    .maxResults(maxResults)
                    .nextToken(paginationToken)
                    .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels= labelDetectionResult.labels();
                for (LabelDetection detectedLabel: detectedLabels) {
                    long seconds=detectedLabel.timestamp();
                    Label label=detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() + ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult !=null && labelDetectionResult.nextToken() !=
null);

        } catch(RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```
   public static void startModerationDetection(RekognitionClient rekClient,
                                               NotificationChannel channel,
                                               String bucket,
                                               String video) {

       try {
           S3Object s3Obj = S3Object.builder()
               .bucket(bucket)
               .name(video)
               .build();

           Video vidOb = Video.builder()
               .s3Object(s3Obj)
               .build();

           StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
               .jobTag("Moderation")
               .notificationChannel(channel)
               .video(vidOb)
               .build();

           StartContentModerationResponse startModDetectionResult =
rekClient.startContentModeration(modDetectionRequest);
           startJobId=startModDetectionResult.jobId();

       } catch(RekognitionException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void GetModResults(RekognitionClient rekClient) {

       try {
           String paginationToken=null;
           GetContentModerationResponse modDetectionResponse=null;
           boolean finished = false;
           String status;
           int yy=0 ;

           do{
               if (modDetectionResponse !=null)
                   paginationToken = modDetectionResponse.nextToken();

               GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                   .jobId(startJobId)
                   .nextToken(paginationToken)
                   .maxResults(10)
                   .build();

               // Wait until the job succeeds
               while (!finished) {
                   modDetectionResponse = rekClient.getContentModeration(modRequest);
                   status = modDetectionResponse.jobStatusAsString();

                   if (status.compareTo("SUCCEEDED") == 0)
                       finished = true;
                   else {
                       System.out.println(yy + " status is: " + status);
                       Thread.sleep(1000);
                   }
                   yy++;
               }
```

```
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null
                VideoMetadata videoMetaData=modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod: mods) {
                    long seconds=mod.timestamp()/1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse !=null && modDetectionResponse.nextToken() !=
null);

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

Detect technical cue segments and shot detection segments in a video stored in an Amazon S3 bucket.

```
    public static void StartSegmentDetection (RekognitionClient rekClient,
                                  NotificationChannel channel,
                                  String bucket,
                                  String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidOb = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                .minSegmentConfidence(60F)
                .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                .minSegmentConfidence(60F)
                .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                .shotFilter(cueDetectionFilter)
                .technicalCueFilter(technicalCueDetectionFilter)
                .build();
```

```
            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                .jobTag("DetectingLabels")
                .notificationChannel(channel)
                .segmentTypes(SegmentType.TECHNICAL_CUE , SegmentType.SHOT)
                .video(vidOb)
                .filters(filters)
                .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();

        } catch(RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {

        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }
```

```
                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
                    System.out.println("\tDuration : " + seconds + " milliseconds");
                    System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse !=null && segDetectionResponse.nextToken() !=
null);

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

Detect text in a video stored in a video stored in an Amazon S3 bucket.

```
    public static void startTextLabels(RekognitionClient rekClient,
                                       NotificationChannel channel,
                                       String bucket,
                                       String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidOb = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                .jobTag("DetectingLabels")
                .notificationChannel(channel)
                .video(vidOb)
                .build();
```

```
            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void GetTextResults(RekognitionClient rekClient) {

        try {
            String paginationToken=null;
            GetTextDetectionResponse textDetectionResponse=null;
            boolean finished = false;
            String status;
            int yy=0 ;

            do{
                if (textDetectionResponse !=null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                    .jobId(startJobId)
                    .nextToken(paginationToken)
                    .maxResults(10)
                    .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText: labels) {
                    System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " + detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                    System.out.println("Type: " + detectedText.textDetection().type());
                    System.out.println("Text: " +
detectedText.textDetection().detectedText());
```

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

            } while (textDetectionResponse !=null &&
textDetectionResponse.nextToken() != null);

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

Detect people in a video stored in a video stored in an Amazon S3 bucket.

```
    public static void startPersonLabels(RekognitionClient rekClient,
                                         NotificationChannel channel,
                                         String bucket,
                                         String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidOb = Video.builder()
                .s3Object(s3Obj)
                .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                .jobTag("DetectingLabels")
                .video(vidOb)
                .notificationChannel(channel)
                .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch(RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void GetPersonDetectionResults(RekognitionClient rekClient) {

        try {
            String paginationToken=null;
            GetPersonTrackingResponse personTrackingResult=null;
            boolean finished = false;
            String status;
            int yy=0 ;

            do{
                if (personTrackingResult !=null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();
```

```
                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null
                VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<PersonDetection> detectedPersons= personTrackingResult.persons();
                for (PersonDetection detectedPerson: detectedPersons) {

                    long seconds=detectedPerson.timestamp()/1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult !=null && personTrackingResult.nextToken() !=
null);

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection

- StartTextDetection

Route 53 domain registration examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Route 53 domain registration.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Route 53 domain registration

The following code examples show how to get started using Route 53 domain registration.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the “com” type, and
 displays prices for registration and renewal.
 *
 */
public class Route53Hello {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <domainType> \n\n" +
            "Where:\n" +
            "    domainType - The domain type (for example, com). \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
```

```
        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
 domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                .maxItems(10)
                .tld(domainType)
                .build();

            ListPricesResponse response =
 route53DomainsClient.listPrices(pricesRequest);
            List<DomainPrice> prices = response.prices();
            for (DomainPrice pr: prices) {
                System.out.println("Name: "+pr.name());
                System.out.println("Registration: "+pr.registrationPrice().price() + "
" +pr.registrationPrice().currency());
                System.out.println("Renewal: "+pr.renewalPrice().price() + " "
+pr.renewalPrice().currency());
                System.out.println("Transfer: "+pr.transferPrice().price() + " "
+pr.transferPrice().currency());
                System.out.println("Change Ownership:
"+pr.changeOwnershipPrice().price() + " " +pr.changeOwnershipPrice().currency());
                System.out.println("Restoration: "+pr.restorationPrice().price() + " "
+pr.restorationPrice().currency());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see ListPrices in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Check domain availability

The following code example shows how to check the availability of a domain.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkDomainAvailability(Route53DomainsClient
 route53DomainsClient, String domainSuggestion) {
```

```
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                .domainName(domainSuggestion)
                .build();

            CheckDomainAvailabilityResponse response =
route53DomainsClient.checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion +" is
"+response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CheckDomainAvailability in *AWS SDK for Java 2.x API Reference*.

Check domain transferability

The following code example shows how to check the transferability of a domain.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion){
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                .domainName(domainSuggestion)
                .build();

            CheckDomainTransferabilityResponse response =
route53DomainsClient.checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability:
"+response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see CheckDomainTransferability in *AWS SDK for Java 2.x API Reference*.

Get domain details

The following code example shows how to get the details for a domain.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion){
       try {
           GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
               .domainName(domainSuggestion)
               .build();

           GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
           System.out.println("The contact first name is " +
response.registrantContact().firstName());
           System.out.println("The contact last name is " +
response.registrantContact().lastName());
           System.out.println("The contact org name is " +
response.registrantContact().organizationName());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see GetDomainDetail in *AWS SDK for Java 2.x API Reference*.

Get operation details

The following code example shows how to get details on an operation.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getOperationalDetail(Route53DomainsClient route53DomainsClient,
String operationId) {
       try {
           GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
               .operationId(operationId)
               .build();

           GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
           System.out.println("Operation detail message is "+response.message());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see GetOperationDetail in *AWS SDK for Java 2.x API Reference*.

Get suggested domain names

The following code example shows how to get domain name suggestions.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listDomainSuggestions(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
       try {
           GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
               .domainName(domainSuggestion)
               .suggestionCount(5)
               .onlyAvailable(true)
               .build();

           GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
           List<DomainSuggestion> suggestions = response.suggestionsList();
           for (DomainSuggestion suggestion: suggestions) {
               System.out.println("Suggestion Name: "+suggestion.domainName());
               System.out.println("Availability: "+suggestion.availability());
               System.out.println(" ");
           }

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see GetDomainSuggestions in *AWS SDK for Java 2.x API Reference*.

List domain prices

The following code example shows how to list domain prices.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
       try {
           ListPricesRequest pricesRequest = ListPricesRequest.builder()
               .tld(domainType)
               .build();

           ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
           listRes.stream()
               .flatMap(r -> r.prices().stream())
               .forEach(content -> System.out.println(" Name: " + content.name() +
                   " Registration: " + content.registrationPrice().price() + " " +
content.registrationPrice().currency() +
                   " Renewal: "+ content.renewalPrice().price() + " " +
content.renewalPrice().currency() ));

       } catch (Route53Exception e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListPrices in *AWS SDK for Java 2.x API Reference*.

List domains

The following code example shows how to list the registered domains.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes = route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                .flatMap(r -> r.domains().stream())
                .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListDomains in *AWS SDK for Java 2.x API Reference*.

List operations

The following code example shows how to list operations.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest = ListOperationsRequest.builder()
                .submittedSince(myTime)
                .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
```

```
            listRes.stream()
                .flatMap(r -> r.operations().stream())
                .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                    " Status: " + content.statusAsString() +
                    " Date: "+content.submittedDate()));


        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListOperations in *AWS SDK for Java 2.x API Reference*.

Register a domain

The following code example shows how to register a domain.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
                                                   String domainSuggestion,
                                                   String phoneNumber,
                                                   String email,
                                                   String firstName,
                                                   String lastName,
                                                   String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                .contactType(ContactType.COMPANY)
                .state("LA")
                .countryCode(CountryCode.IN)
                .email(email)
                .firstName(firstName)
                .lastName(lastName)
                .city(city)
                .phoneNumber(phoneNumber)
                .organizationName("My Org")
                .addressLine1("My Address")
                .zipCode("123 123")
                .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                .adminContact(contactDetail)
                .registrantContact(contactDetail)
                .techContact(contactDetail)
                .domainName(domainSuggestion)
                .autoRenew(true)
                .durationInYears(1)
                .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
```

```
            System.out.println("Registration requested. Operation Id: "
+response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see RegisterDomain in *AWS SDK for Java 2.x API Reference*.

View billing

The following code example shows how to view billing records.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listBillingRecords(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                .start(myStartTime)
                .end(myEndTime)
                .build();

            ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                .flatMap(r -> r.billingRecords().stream())
                .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                    " Operation: " + content.operationAsString() +
                    " Price: "+content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ViewBilling in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with domains

The following code example shows how to:

- List current domains, and list operations in the past year.
- View billing for the past year, and view prices for domain types.
- Get domain suggestions.
- Check domain availability and transferability.
- Optionally, request a domain registration.
- Get an operation detail.
- Optionally, get a domain detail.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list domains,
 the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
 <lastName> <city>\n\n" +
            "Where:\n" +
            "    domainType - The domain type (for example, com). \n" +
```

```
            "    phoneNumber - The phone number to use (for example, +91.9966564xxx)
"+
            "    email - The email address to use.  "+
            "    domainSuggestion - The domain suggestion (for example,
findmy.accountants). \n" +
            "    firstName - The first name to use to register a domain. \n" +
            "    lastName -  The last name to use to register a domain. \n" +
            "    city - the city to use to register a domain. ";

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        String phoneNumber = args[1];
        String email = args[2] ;
        String domainSuggestion = args[3] ;
        String firstName = args[4] ;
        String lastName = args[5] ;
        String city = args[6] ;
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Route 53 domains example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. List current domains.");
        listDomains(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List operations in the past year.");
        listOperations(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. View billing for the account in the past year.");
        listBillingRecords(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. View prices for domain types.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get domain suggestions.");
        listDomainSuggestions(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Check domain availability.");
        checkDomainAvailability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient, domainSuggestion,
phoneNumber, email, firstName, lastName, city);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get details.");
        System.out.println("Otherwise, an exception is thrown that states " );
        System.out.println("Domain xxxxxxx not found in xxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion){
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                .domainName(domainSuggestion)
                .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient route53DomainsClient,
String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                .operationId(operationId)
                .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is "+response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
                                                   String domainSuggestion,
                                                   String phoneNumber,
                                                   String email,
                                                   String firstName,
```

```
                                                    String lastName,
                                                    String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                .contactType(ContactType.COMPANY)
                .state("LA")
                .countryCode(CountryCode.IN)
                .email(email)
                .firstName(firstName)
                .lastName(lastName)
                .city(city)
                .phoneNumber(phoneNumber)
                .organizationName("My Org")
                .addressLine1("My Address")
                .zipCode("123 123")
                .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                .adminContact(contactDetail)
                .registrantContact(contactDetail)
                .techContact(contactDetail)
                .domainName(domainSuggestion)
                .autoRenew(true)
                .durationInYears(1)
                .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: "
+response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion){
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                .domainName(domainSuggestion)
                .build();

            CheckDomainTransferabilityResponse response =
route53DomainsClient.checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability:
"+response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                .domainName(domainSuggestion)
                .build();
```

```
            CheckDomainAvailabilityResponse response =
route53DomainsClient.checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion +" is
"+response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomainSuggestions(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                .domainName(domainSuggestion)
                .suggestionCount(5)
                .onlyAvailable(true)
                .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion: suggestions) {
                System.out.println("Suggestion Name: "+suggestion.domainName());
                System.out.println("Availability: "+suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                .tld(domainType)
                .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                .flatMap(r -> r.prices().stream())
                .forEach(content -> System.out.println(" Name: " + content.name() +
                    " Registration: " + content.registrationPrice().price() + " " +
content.registrationPrice().currency() +
                    " Renewal: "+ content.renewalPrice().price() + " " +
content.renewalPrice().currency() ));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listBillingRecords(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
```

```
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                .start(myStartTime)
                .end(myEndTime)
                .build();

            ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                .flatMap(r -> r.billingRecords().stream())
                .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                    " Operation: " + content.operationAsString() +
                    " Price: "+content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest = ListOperationsRequest.builder()
                .submittedSince(myTime)
                .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                .flatMap(r -> r.operations().stream())
                .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                    " Status: " + content.statusAsString() +
                    " Date: "+content.submittedDate()));


        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes = route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                .flatMap(r -> r.domains().stream())
                .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Amazon S3 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon S3.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Add CORS rules to a bucket

The following code example shows how to add cross-origin resource sharing (CORS) rules to an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
       try {
           DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
               .bucket(bucketName)
               .expectedBucketOwner(accountId)
               .build();

           s3.deleteBucketCors(bucketCorsRequest) ;
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName, String
accountId) {

        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse = s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule: corsRules) {
                System.out.println("allowOrigins: "+rule.allowedOrigins());
                System.out.println("AllowedMethod: "+rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {

        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                .allowedMethods(allowMethods)
                .allowedOrigins(allowOrigins)
                .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                .corsRules(corsRules)
                .build();

            PutBucketCorsRequest putBucketCorsRequest = PutBucketCorsRequest.builder()
                .bucket(bucketName)
                .corsConfiguration(configuration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutBucketCors in *AWS SDK for Java 2.x API Reference*.

Add a lifecycle configuration to a bucket

The following code example shows how to add a lifecycle configuration to an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {

       try {
           // Create a rule to archive objects with the "glacierobjects/" prefix to
Amazon S3 Glacier.
           LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
               .prefix("glacierobjects/")
               .build();

           Transition transition = Transition.builder()
               .storageClass(TransitionStorageClass.GLACIER)
               .days(0)
               .build();

           LifecycleRule rule1 = LifecycleRule.builder()
               .id("Archive immediately rule")
               .filter(ruleFilter)
               .transitions(transition)
               .status(ExpirationStatus.ENABLED)
               .build();

           // Create a second rule.
           Transition transition2 = Transition.builder()
               .storageClass(TransitionStorageClass.GLACIER)
               .days(0)
               .build();

           List<Transition> transitionList = new ArrayList<>();
           transitionList.add(transition2);

           LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
               .prefix("glacierobjects/")
               .build();

           LifecycleRule rule2 = LifecycleRule.builder()
               .id("Archive and then delete rule")
               .filter(ruleFilter2)
               .transitions(transitionList)
               .status(ExpirationStatus.ENABLED)
               .build();

           // Add the LifecycleRule objects to an ArrayList.
           ArrayList<LifecycleRule> ruleList = new ArrayList<>();
           ruleList.add(rule1);
           ruleList.add(rule2);

           BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
```

```
                .rules(ruleList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Retrieve the configuration and add a new rule.
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId){

        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest =
GetBucketLifecycleConfigurationRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketLifecycleConfigurationResponse response =
s3.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule: rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag predicate.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
```

```
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the configuration from the Amazon S3 bucket.
    public static void deleteLifecycleConfig(S3Client s3, String bucketName, String
accountId) {

        try {
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutBucketLifecycleConfiguration in *AWS SDK for Java 2.x API Reference*.

Add a policy to a bucket

The following code example shows how to add a policy to an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void setPolicy(S3Client s3, String bucketName, String policyText) {

        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }

        System.out.println("Done!");
    }

    // Loads a JSON-formatted policy from a file
    public static String getBucketPolicyFromFile(String policyFile) {

        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
```

- For API details, see PutBucketPolicy in *AWS SDK for Java 2.x API Reference*.

Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Copy an object using an S3Client.

```
    public static String copyBucketObject (S3Client s3, String fromBucket, String
objectKey, String toBucket) {

        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

Use an S3TransferManager to copy an object from one bucket to another. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String bucketName,
                             String key, String destinationBucket, String
 destinationKey){
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
            .sourceBucket(bucketName)
            .sourceKey(key)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        CopyRequest copyRequest = CopyRequest.builder()
            .copyObjectRequest(copyObjectRequest)
            .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- For API details, see CopyObject in *AWS SDK for Java 2.x API Reference*.

Create a bucket

The following code example shows how to create an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createBucket( S3Client s3Client, String bucketName) {

        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
```

```
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName +" is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see CreateBucket in *AWS SDK for Java 2.x API Reference*.

Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {

        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
           .bucket(bucketName)
           .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteBucketPolicy in *AWS SDK for Java 2.x API Reference*.

Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        s3.deleteBucket(deleteBucketRequest);
        s3.close();
```

- For API details, see DeleteBucket in *AWS SDK for Java 2.x API Reference*.

Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteBucketObjects(S3Client s3, String bucketName) {

        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();

            putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(del)
                .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

- For API details, see DeleteObjects in *AWS SDK for Java 2.x API Reference*.

Delete the website configuration from a bucket

The following code example shows how to delete the website configuration from an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteBucketWebsiteConfig(S3Client s3,String bucketName ) {

        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
```

- For API details, see DeleteBucketWebsite in *AWS SDK for Java 2.x API Reference*.

Determine the existence and content type of an object

The following code example shows how to determine the existence and content type of an object in an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Determine the content type of an object.

```
    public static void getContentType (S3Client s3, String bucketName, String keyName)
{

        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
```

```
            String type = objectHead.contentType();
            System.out.println("The object content type is "+type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Get the restore status of an object.

```
    public static void checkStatus(S3Client s3, String bucketName, String keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is
 "+response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see HeadObject in *AWS SDK for Java 2.x API Reference*.

Download objects to a local directory

The following code example shows how to download all objects in an Amazon Simple Storage Service (Amazon S3) bucket to a local directory.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use an S3TransferManager to download all S3 objects in the same S3 bucket. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
```

```
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
                                              String destinationPath, String
 bucketName) {
        DirectoryDownload directoryDownload =
            transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPath))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers().forEach(fail ->
            logger.warn("Object [{}] failed to transfer", fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- For API details, see DownloadDirectory in *AWS SDK for Java 2.x API Reference*.

Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Read data as a byte array using an S3Client.

```
    public static void getObjectBytes (S3Client s3, String bucketName, String keyName,
 String path) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path );
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Use an S3TransferManager to download an object in an S3 bucket to a local file. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest =
            DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
        logger.info("Content length [{}]", downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Read tags that belong to an object using an S3Client.

```
    public static void listTags(S3Client s3, String bucketName, String keyName) {

        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet= tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Get a URL for an object using an S3Client.

```
    public static void getURL(S3Client s3, String bucketName, String keyName ) {
```

```
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  "+keyName +" is "+ url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Get an object by using the S3Presigner client object using an S3Client.

```
      public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName ) {

          try {
              GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                  .bucket(bucketName)
                  .key(keyName)
                  .build();

              GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                  .signatureDuration(Duration.ofMinutes(60))
                  .getObjectRequest(getObjectRequest)
                  .build();

              PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
              String theUrl = presignedGetObjectRequest.url().toString();
              System.out.println("Presigned URL: " + theUrl);
              HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
              presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
               values.forEach(value -> {
                   connection.addRequestProperty(header, value);
                });
              });

              // Send any request payload that the service needs (not needed when
isBrowserExecutable is true).
              if (presignedGetObjectRequest.signedPayload().isPresent()) {
                  connection.setDoOutput(true);

              try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                   OutputStream httpOutputStream = connection.getOutputStream()) {
                   IoUtils.copy(signedPayload, httpOutputStream);
              }
          }

          // Download the result of executing the request.
          try (InputStream content = connection.getInputStream()) {
              System.out.println("Service returned response: ");
              IoUtils.copy(content, System.out);
          }
```

```
        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
```

- For API details, see GetObject in *AWS SDK for Java 2.x API Reference*.

Get the ACL of a bucket

The following code example shows how to get the access control list (ACL) of an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String getBucketACL(S3Client s3, String objectKey,String bucketName)
{

        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- For API details, see GetBucketAcl in *AWS SDK for Java 2.x API Reference*.

Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String getPolicy(S3Client s3, String bucketName) {
```

```
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- For API details, see GetBucketPolicy in *AWS SDK for Java 2.x API Reference*.

List in-progress multipart uploads

The following code example shows how to list in-progress multipart uploads to an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listUploads( S3Client s3, String bucketName) {

        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

            ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload: uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key() + "\",
id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListMultipartUploads in *AWS SDK for Java 2.x API Reference*.

List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listBucketObjects(S3Client s3, String bucketName ) {

        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                .builder()
                .bucket(bucketName)
                .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    //convert bytes to kbs.
    private static long calKb(Long val) {
        return val/1024;
    }
```

List objects using pagination.

```
    public static void listBucketObjects(S3Client s3, String bucketName ) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() + "
 size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListObjects in *AWS SDK for Java 2.x API Reference*.

Restore an archived copy of an object

The following code example shows how to restore an archived copy of an object back into an S3 bucket.

SDK for Java 2.x

Note

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

```
   public static void restoreS3Object(S3Client s3, String bucketName, String keyName,
String expectedBucketOwner) {

       try {
           RestoreRequest restoreRequest = RestoreRequest.builder()
               .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
               .build();

           RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
               .expectedBucketOwner(expectedBucketOwner)
               .bucket(bucketName)
               .key(keyName)
               .restoreRequest(restoreRequest)
               .build();

           s3.restoreObject(objectRequest);

       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- For API details, see RestoreObject in *AWS SDK for Java 2.x API Reference*.

Set a new ACL for a bucket

The following code example shows how to set a new access control list (ACL) for an S3 bucket.

SDK for Java 2.x

Note

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

```
   public static void setBucketAcl(S3Client s3, String bucketName, String id) {

       try {
           Grant ownerGrant = Grant.builder()
               .grantee(builder -> builder.id(id)
               .type(Type.CANONICAL_USER))
                   .permission(Permission.FULL_CONTROL)
                   .build();

           List<Grant> grantList2 = new ArrayList<>();
           grantList2.add(ownerGrant);

           AccessControlPolicy acl = AccessControlPolicy.builder()
               .owner(builder -> builder.id(id))
               .grants(grantList2)
               .build();

           PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
```

```
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
```

- For API details, see PutBucketAcl in *AWS SDK for Java 2.x API Reference*.

Set the website configuration for a bucket

The following code example shows how to set the website configuration for an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void setWebsiteConfig( S3Client s3, String bucketName, String
indexDoc) {

        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq = PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutBucketWebsite in *AWS SDK for Java 2.x API Reference*.

Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Upload a file to a bucket using an S3Client.

```
   public static String putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {

       try {
           Map<String, String> metadata = new HashMap<>();
           metadata.put("x-amz-meta-myVal", "test");
           PutObjectRequest putOb = PutObjectRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .metadata(metadata)
               .build();

           PutObjectResponse response = s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));
           return response.eTag();

       } catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }

       return "";
   }

   // Return a byte array.
   private static byte[] getObjectFile(String filePath) {

       FileInputStream fileInputStream = null;
       byte[] bytesArray = null;

       try {
           File file = new File(filePath);
           bytesArray = new byte[(int) file.length()];
           fileInputStream = new FileInputStream(file);
           fileInputStream.read(bytesArray);

       } catch (IOException e) {
           e.printStackTrace();
       } finally {
           if (fileInputStream != null) {
               try {
                   fileInputStream.close();
               } catch (IOException e) {
                   e.printStackTrace();
               }
           }
       }

       return bytesArray;
   }
```

Use an S3TransferManager to upload a file to a bucket. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.net.URL;
import java.nio.file.Paths;
```

```
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, String filePath) {
        UploadFileRequest uploadFileRequest =
            UploadFileRequest.builder()
                .putObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())
                .source(Paths.get(filePath))
                .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Upload an object to a bucket and set tags using an S3Client.

```
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
 objectKey, String objectPath) {

        try {

            Tag tag1 = Tag.builder()
                .key("Tag 1")
                .value("This is tag 1")
                .build();

            Tag tag2 = Tag.builder()
                .key("Tag 2")
                .value("This is tag 2")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(allTags)
                .build();

            s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateObjectTags(S3Client s3, String bucketName, String
 objectKey) {

        try {
            GetObjectTaggingRequest taggingRequest = GetObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();
```

```
            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag: obTags) {
                System.out.println("The tag key is: "+sinTag.key());
                System.out.println("The tag value is: "+sinTag.value());
            }

            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                .key("Tag 3")
                .value("This is tag 3")
                .build();

            Tag tag4 = Tag.builder()
                .key("Tag 4")
                .value("This is tag 4")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectTaggingRequest taggingRequest1 = PutObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(updatedTags)
                .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag: modTags) {
                System.out.println("The tag key is: "+sinTag.key());
                System.out.println("The tag value is: "+sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Upload an object to a bucket and set metadata using an S3Client.

```
    public static String putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {

        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();
```

```
            PutObjectResponse response = s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));
            return response.eTag();

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        return "";
    }

    // Return a byte array.
    private static byte[] getObjectFile(String filePath) {

        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
```

Upload an object to a bucket and set an object retention value using an S3Client.

```
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {

        try{
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                .mode("COMPLIANCE")
                .retainUntilDate(instant)
                .build();

            PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
                .bucket(bucket)
                .key(key)
                .bypassGovernanceRetention(true)
                .retention(lockRetention)
                .build();

            // To set Retention on an object, the Amazon S3 bucket must support object
locking, otherwise an exception is thrown.
```

```
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully placed
 on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutObject in *AWS SDK for Java 2.x API Reference*.

Upload directory to a bucket

The following code example shows how to upload a local directory recursively to an Amazon Simple Storage Service (Amazon S3) bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Use an S3TransferManager to upload a local directory. View the complete file and test.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
                                   String sourceDirectory, String bucketName){
        DirectoryUpload directoryUpload =
            transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
 directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers().forEach(fail ->
            logger.warn("Object [{}] failed to transfer", fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- For API details, see UploadDirectory in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a presigned URL

The following code example shows how to create a presigned URL for Amazon S3 and upload an object.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void signBucket(S3Presigner presigner, String bucketName, String
keyName) {

       try {
           PutObjectRequest objectRequest = PutObjectRequest.builder()
               .bucket(bucketName)
               .key(keyName)
               .contentType("text/plain")
               .build();

           PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
               .signatureDuration(Duration.ofMinutes(10))
               .putObjectRequest(objectRequest)
               .build();

           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           System.out.println("Presigned URL to upload a file to: " +myURL);
           System.out.println("Which HTTP method needs to be used when uploading a
file: " + presignedRequest.httpRequest().method());

           // Upload content to the Amazon S3 bucket by using this URL.
           URL url = presignedRequest.url();

           // Create the connection and use it to upload the new object by using the
presigned URL.
           HttpURLConnection connection = (HttpURLConnection) url.openConnection();
           connection.setDoOutput(true);
           connection.setRequestProperty("Content-Type","text/plain");
           connection.setRequestMethod("PUT");
           OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
           out.write("This text was uploaded as an object by using a presigned URL.");
           out.close();

           connection.getResponseCode();
           System.out.println("HTTP response code is " +
connection.getResponseCode());

       } catch (S3Exception | IOException e) {
           e.getStackTrace();
       }
   }
```

Get started with buckets and objects

The following code example shows how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the bucket objects and the bucket.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 *  Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  This Java code example performs the following tasks:
 *
 *  1. Creates an Amazon S3 bucket.
 *  2. Uploads an object to the bucket.
 *  3. Downloads the object to another local file.
 *  4. Uploads an object using multipart upload.
 *  5. List all objects located in the Amazon S3 bucket.
 *  6. Copies the object to another Amazon S3 bucket.
 *  7. Deletes the object from the Amazon S3 bucket.
 *  8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws IOException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <bucketName> <key> <objectPath> <savePath> <toBucket>\n\n" +
            "Where:\n" +
            "    bucketName - The Amazon S3 bucket to create.\n\n" +
            "    key - The key to use.\n\n" +
            "    objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf). "+
            "    savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf). " +
            "    toBucket - An Amazon S3 bucket to where an object is copied to (for
 example, C:/AWS/book2.pdf). ";

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4] ;

        ProfileCredentialsProvider credentialsProvider =
 ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 example scenario.");
```

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

        System.out.println(DASHES);
        System.out.println("1. Create an Amazon S3 bucket.");
        createBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update a local file to the Amazon S3 bucket.");
        uploadLocalFile(s3, bucketName, key, objectPath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Download the object to another local file.");
        getObjectBytes (s3, bucketName, key, savePath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Perform a multipart upload.");
        String multipartKey = "multiPartKey";
        multipartUpload(s3, toBucket, multipartKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List all objects located in the Amazon S3 bucket.");
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName) ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
        copyBucketObject (s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object
    public static void createBucket( S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName +" is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucket(S3Client client, String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        client.deleteBucket(deleteBucketRequest);
        System.out.println(bucket +" was deleted.");
    }

    /**
     * Upload an object in parts
     */
    private static void multipartUpload(S3Client s3, String bucketName, String key) {
        int mB = 1024 * 1024;
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB))).eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
            .uploadId(uploadId)
            .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB))).eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(part1, part2)
            .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
            .bucket(bucketName)
```

```
            .key(key)
            .uploadId(uploadId)
            .multipartUpload(completedMultipartUpload)
            .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
        return ByteBuffer.wrap(b);
    }

    // Return a byte array
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return bytesArray;
    }

    public static void getObjectBytes (S3Client s3, String bucketName, String keyName,
String path ) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       s3.putObject(objectRequest, RequestBody.fromBytes(getObjectFile(objectPath)));
   }

   public static void listAllObjects(S3Client s3, String bucketName) {

       ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .maxKeys(1)
           .build();

       boolean done = false;
       while (!done) {
           ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
           for (S3Object content : listObjResponse.contents()) {
               System.out.println(content.key());
           }

           if (listObjResponse.nextContinuationToken() == null) {
               done = true;
           }

           listObjectsReqManual = listObjectsReqManual.toBuilder()
               .continuationToken(listObjResponse.nextContinuationToken())
               .build();
       }
   }

   public static void anotherListExample(S3Client s3, String bucketName) {

      ListObjectsV2Request listReq = ListObjectsV2Request.builder()
          .bucket(bucketName)
          .maxKeys(1)
          .build();

      ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

      // Process response pages
      listRes.stream()
          .flatMap(r -> r.contents().stream())
          .forEach(content -> System.out.println(" Key: " + content.key() + " size = "
+ content.size()));

       // Helper method to work with paginated collection of items directly
       listRes.contents().stream()
           .forEach(content -> System.out.println(" Key: " + content.key() + " size =
" + content.size()));

       for (S3Object content : listRes.contents()) {
           System.out.println(" Key: " + content.key() + " size = " + content.size());
       }
   }


   public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
```

```
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key +" was deleted");
    }

    public static String copyBucketObject (S3Client s3, String fromBucket, String
 objectKey, String toBucket) {

        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
 StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .copySource(encodedUrl)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The "+ objectKey +" was copied to "+toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjects
  - PutObject

S3 Glacier examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with S3 Glacier.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a vault

The following code example shows how to create an Amazon S3 Glacier vault.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void createGlacierVault(GlacierClient glacier, String vaultName ) {

        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            CreateVaultResponse createVaultResult = glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
createVaultResult.location());

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

- For API details, see CreateVault in *AWS SDK for Java 2.x API Reference*.

Delete a vault

The following code example shows how to delete an Amazon S3 Glacier vault.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {

        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                .vaultName(vaultName)
                .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");

        } catch(GlacierException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see DeleteVault in *AWS SDK for Java 2.x API Reference*.

Delete an archive

The following code example shows how to delete an Amazon S3 Glacier archive.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
 String accountId, String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                .vaultName(vaultName)
                .accountId(accountId)
                .archiveId(archiveId)
                .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The vault was deleted!");

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

- For API details, see DeleteArchive in *AWS SDK for Java 2.x API Reference*.

List vaults

The following code example shows how to list Amazon S3 Glacier vaults.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllVault(GlacierClient glacier) {

        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");

        try {

            while (!listComplete) {
                ListVaultsResponse response = null;
```

```
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                        .marker(newMarker)
                        .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                        .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v: vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

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

- For API details, see ListVaults in *AWS SDK for Java 2.x API Reference*.

Retrieve a vault inventory

The following code example shows how to retrieve an Amazon S3 Glacier vault inventory.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {

       try {

           JobParameters job = JobParameters.builder()
               .type("inventory-retrieval")
               .build();

           InitiateJobRequest initJob = InitiateJobRequest.builder()
               .jobParameters(job)
               .accountId(accountId)
               .vaultName(vaultName)
               .build();
```

```
            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " +response.jobId()) ;
            System.out.println("The relative URI path of the job is: "
+response.location()) ;
            return response.jobId();

        } catch(GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);

        }
        return "";
    }

    //  Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {

       try{
           boolean finished = false;
           String jobStatus;
           int yy=0;

           while (!finished) {
               DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                   .jobId(jobId)
                   .accountId(account)
                   .vaultName(name)
                   .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

               if (jobStatus.compareTo("Succeeded") == 0)
                   finished = true;
               else {
                   System.out.println(yy + " status is: " + jobStatus);
                   Thread.sleep(1000);
               }
               yy++;
           }

           System.out.println("Job has Succeeded");
           GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
               .jobId(jobId)
               .vaultName(name)
               .accountId(account)
               .build();

           ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
           // Write the data to a local file.
           byte[] data = objectBytes.asByteArray();
           File myFile = new File(path);
           OutputStream os = new FileOutputStream(myFile);
           os.write(data);
           System.out.println("Successfully obtained bytes from a Glacier vault");
           os.close();

       } catch(GlacierException | InterruptedException | IOException e) {
           System.out.println(e.getMessage());
           System.exit(1);

       }
    }
```

- For API details, see InitiateJob in *AWS SDK for Java 2.x API Reference*.

Upload an archive to a vault

The following code example shows how to upload an archive to an Amazon S3 Glacier vault.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {

       // Get an SHA-256 tree hash value.
       String checkVal = computeSHA256(myFile);
       try {
           UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
               .vaultName(vaultName)
               .checksum(checkVal)
               .build();

           UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
           return res.archiveId();

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

   private static String computeSHA256(File inputFile) {

       try {
           byte[] treeHash = computeSHA256TreeHash(inputFile);
           System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
           return toHex(treeHash);

       } catch (IOException ioe) {
           System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
           System.exit(-1);

       } catch (NoSuchAlgorithmException nsae) {
           System.err.format("Cannot locate MessageDigest algorithm for SHA-256: %s",
nsae.getMessage());
           System.exit(-1);
       }
       return "";
   }

   public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
           NoSuchAlgorithmException {

       byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
       return computeSHA256TreeHash(chunkSHA256Hashes);
   }

   /**
```

```
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {
```

```
                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
```

- For API details, see UploadArchive in *AWS SDK for Java 2.x API Reference*.

Amazon SES examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SES.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

List email templates

The following code example shows how to list Amazon SES email templates.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllTemplates(SesV2Client sesv2Client) {

        try {
            ListEmailTemplatesRequest templatesRequest =
ListEmailTemplatesRequest.builder()
                .pageSize(1)
                .build();

            ListEmailTemplatesResponse response =
sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata().forEach(template ->
                    System.out.println("Template name: " + template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListTemplates in *AWS SDK for Java 2.x API Reference*.

List identities

The following code example shows how to list Amazon SES identities.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listSESIdentities(SesClient client) {

        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity: identities) {
                System.out.println("The identity is "+identity);
            }

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

- For API details, see ListIdentities in *AWS SDK for Java 2.x API Reference*.

Send email

The following code example shows how to send email with Amazon SES.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void send(SesClient client,
                            String sender,
                            String recipient,
                            String subject,
                            String bodyHTML
    ) throws MessagingException {

        Destination destination = Destination.builder()
            .toAddresses(recipient)
            .build();

        Content content = Content.builder()
            .data(bodyHTML)
            .build();

        Content sub = Content.builder()
            .data(subject)
            .build();

        Body body = Body.builder()
            .html(content)
            .build();

        Message msg = Message.builder()
            .subject(sub)
            .body(body)
            .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
            .destination(destination)
            .message(msg)
            .source(sender)
            .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendemailAttachment(SesClient client,
                            String sender,
                            String recipient,
                            String subject,
                            String bodyText,
                            String bodyHTML,
                            String fileLocation) throws AddressException,
MessagingException, IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());
```

```
        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
 InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();

        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");

        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);

        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent, "application/
vnd.openxmlformats-officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));

        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);

            SdkBytes data = SdkBytes.fromByteArray(arr);
            RawMessage rawMessage = RawMessage.builder()
                .data(data)
                .build();
```

```
            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                .rawMessage(rawMessage)
                .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
```

- For API details, see SendEmail in *AWS SDK for Java 2.x API Reference*.

Send templated email

The following code example shows how to send templated email with Amazon SES.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void send(SesV2Client client, String sender, String recipient, String
templateName){

       Destination destination = Destination.builder()
           .toAddresses(recipient)
           .build();

       /*
        Specify both name and favorite animal (favoriteanimal) in your code when
defining the Template object.
        If you don't specify all the variables in the template, Amazon SES doesn't
send the email.
       */
       Template myTemplate = Template.builder()
           .templateName(templateName)
           .templateData("{\n" +
             "  \"name\": \"Jason\"\n," +
             "  \"favoriteanimal\": \"Cat\"\n" +
             "}")
           .build();

       EmailContent emailContent = EmailContent.builder()
           .template(myTemplate)
           .build();

       SendEmailRequest emailRequest = SendEmailRequest.builder()
           .destination(destination)
           .content(emailContent)
           .fromEmailAddress(sender)
           .build();

       try {
           System.out.println("Attempting to send an email based on a template using
the AWS SDK for Java (v2)...");
           client.sendEmail(emailRequest);
           System.out.println("email based on a template was sent");
```

```
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see SendTemplatedEmail in *AWS SDK for Java 2.x API Reference*.

Amazon SES API v2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SES API v2.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Send an email

The following code example shows how to send an Amazon SES API v2 email.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Sends a message.

```
        public static void send(SesV2Client client,
                                String sender,
                                String recipient,
                                String subject,
                                String bodyHTML
        ){

            Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

            Content content = Content.builder()
                .data(bodyHTML)
                .build();

            Content sub = Content.builder()
                .data(subject)
                .build();

            Body body = Body.builder()
                .html(content)
```

```
                .build();

            Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(msg)
                 .build();

            SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

            try {
                System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
                client.sendEmail(emailRequest);
                System.out.println("email was sent");

            } catch (SesV2Exception e) {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
```

- For API details, see SendEmail in *AWS SDK for Java 2.x API Reference*.

Amazon SNS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SNS.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Add tags to a topic

The following code example shows how to add tags to an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void addTopicTags(SnsClient snsClient, String topicArn) {

        try {
            Tag tag = Tag.builder()
                .key("Team")
                .value("Development")
                .build();

            Tag tag2 = Tag.builder()
                .key("Environment")
                .value("Gamma")
                .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .resourceArn(topicArn)
                .tags(tagList)
                .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to "+topicArn);

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

- For API details, see TagResource in *AWS SDK for Java 2.x API Reference*.

Check whether a phone number is opted out

The following code example shows how to check whether a phone number is opted out of receiving Amazon SNS messages.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void checkPhone(SnsClient snsClient, String phoneNumber) {

        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
CheckIfPhoneNumberIsOptedOutRequest.builder()
                .phoneNumber(phoneNumber)
                .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(result.isOptedOut() + "Phone Number " + phoneNumber + "
has Opted Out of receiving sns messages." +
                "\n\nStatus was " + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

- For API details, see CheckIfPhoneNumberIsOptedOut in *AWS SDK for Java 2.x API Reference*.

Confirm an endpoint owner wants to receive messages

The following code example shows how to confirm the owner of an endpoint wants to receive Amazon SNS messages by validating the token sent to the endpoint by an earlier Subscribe action.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void confirmSub(SnsClient snsClient, String subscriptionToken, String
topicArn ) {

       try {
           ConfirmSubscriptionRequest request = ConfirmSubscriptionRequest.builder()
               .token(subscriptionToken)
               .topicArn(topicArn)
               .build();

           ConfirmSubscriptionResponse result =
snsClient.confirmSubscription(request);
           System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n" +
result.subscriptionArn());

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

- For API details, see ConfirmSubscription in *AWS SDK for Java 2.x API Reference*.

Create a topic

The following code example shows how to create an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createSNSTopic(SnsClient snsClient, String topicName ) {

        CreateTopicResponse result = null;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();
```

```
            result = snsClient.createTopic(request);
            return result.topicArn();

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

- For API details, see CreateTopic in *AWS SDK for Java 2.x API Reference*.

Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void unSub(SnsClient snsClient, String subscriptionArn) {

        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
                + "\n\nSubscription was removed for " + request.subscriptionArn());

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

- For API details, see Unsubscribe in *AWS SDK for Java 2.x API Reference*.

Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {

        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
```

```
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

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

- For API details, see DeleteTopic in *AWS SDK for Java 2.x API Reference*.

Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn ) {

        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                .topicArn(topicArn)
                .build();

            GetTopicAttributesResponse result = snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()
+ "\n\nAttributes: \n\n" + result.attributes());

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

- For API details, see GetTopicAttributes in *AWS SDK for Java 2.x API Reference*.

Get the settings for sending SMS messages

The following code example shows how to get the settings for sending Amazon SNS SMS messages.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getSNSAttrutes(SnsClient snsClient,String topicArn ) {

        try {
            GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
```

```
                .subscriptionArn(topicArn)
                .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
```

- For API details, see GetSMSAttributes in *AWS SDK for Java 2.x API Reference*.

List opted out phone numbers

The following code example shows how to list phone numbers that are opted out of receiving Amazon SNS messages.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listOpts( SnsClient snsClient) {

        try {
            ListPhoneNumbersOptedOutRequest request =
ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode() +
"\n\nPhone Numbers: \n\n" + result.phoneNumbers());

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

- For API details, see ListPhoneNumbersOptedOut in *AWS SDK for Java 2.x API Reference*.

List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listSNSSubscriptions( SnsClient snsClient) {

        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see ListSubscriptions in *AWS SDK for Java 2.x API Reference*.

List topics

The following code example shows how to list Amazon SNS topics.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listSNSTopics(SnsClient snsClient) {

        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                   .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopics\n\n" + result.topics());

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

- For API details, see ListTopics in *AWS SDK for Java 2.x API Reference*.

Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
       try {
           PublishRequest request = PublishRequest.builder()
               .message(message)
               .phoneNumber(phoneNumber)
               .build();

           PublishResponse result = snsClient.publish(request);
           System.out.println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

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

- For API details, see Publish in *AWS SDK for Java 2.x API Reference*.

Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void pubTopic(SnsClient snsClient, String message, String topicArn) {

       try {
           PublishRequest request = PublishRequest.builder()
               .message(message)
               .topicArn(topicArn)
               .build();

           PublishResponse result = snsClient.publish(request);
           System.out.println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

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

- For API details, see Publish in *AWS SDK for Java 2.x API Reference*.

Set a filter policy

The following code example shows how to set an Amazon SNS filter policy.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public static void usePolicy(SnsClient snsClient, String subscriptionArn) {

    try {
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
        // Add a filter policy attribute with a single value
        fp.addAttribute("store", "example_corp");
        fp.addAttribute("event", "order_placed");

        // Add a prefix attribute
        fp.addAttributePrefix("customer_interests", "bas");

        // Add an anything-but attribute
        fp.addAttributeAnythingBut("customer_interests", "baseball");

        // Add a filter policy attribute with a list of values
        ArrayList<String> attributeValues = new ArrayList<>();
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);

        // Add a numeric attribute with a range
        fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

        // Apply the filter policy attributes to an Amazon SNS subscription
        fp.apply(snsClient, subscriptionArn);

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

- For API details, see SetSubscriptionAttributes in *AWS SDK for Java 2.x API Reference*.

Set the default settings for sending SMS messages

The following code example shows how to set the default settings for sending SMS messages using Amazon SNS.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class SetSMSAttributes {
```

```
   public static void main(String[] args) {

       HashMap<String, String> attributes = new HashMap<>(1);
       attributes.put("DefaultSMSType", "Transactional");
       attributes.put("UsageReportS3Bucket", "janbucket" );

       SnsClient snsClient = SnsClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(ProfileCredentialsProvider.create())
           .build();
       setSNSAttributes(snsClient, attributes);
       snsClient.close();
   }

   public static void setSNSAttributes( SnsClient snsClient, HashMap<String, String>
attributes) {

       try {
           SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
               .attributes(attributes)
               .build();

           SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
           System.out.println("Set default Attributes to " + attributes + ". Status
was " + result.sdkHttpResponse().statusCode());

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

- For API details, see SetSmsAttributes in *AWS SDK for Java 2.x API Reference*.

Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {

       try {
           SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
               .attributeName(attribute)
               .attributeValue(value)
               .topicArn(topicArn)
               .build();

           SetTopicAttributesResponse result = snsClient.setTopicAttributes(request);
           System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
               + " updated " + request.attributeName() + " to " +
request.attributeValue());

       } catch (SnsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
```

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

- For API details, see SetTopicAttributes in *AWS SDK for Java 2.x API Reference*.

Subscribe a Lambda function to a topic

The following code example shows how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {

       try {
           SubscribeRequest request = SubscribeRequest.builder()
               .protocol("lambda")
               .endpoint(lambdaArn)
               .returnSubscriptionArn(true)
               .topicArn(topicArn)
               .build();

           SubscribeResponse result = snsClient.subscribe(request);
           return result.subscriptionArn();

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

- For API details, see Subscribe in *AWS SDK for Java 2.x API Reference*.

Subscribe an HTTP endpoint to a topic

The following code example shows how to subscribe an HTTP or HTTPS endpoint so it receives notifications from an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void subHTTPS(SnsClient snsClient, String topicArn, String url ) {

       try {
           SubscribeRequest request = SubscribeRequest.builder()
               .protocol("https")
               .endpoint(url)
               .returnSubscriptionArn(true)
```

```
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn() + "\n
\n Status is " + result.sdkHttpResponse().statusCode());

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

- For API details, see Subscribe in *AWS SDK for Java 2.x API Reference*.

Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void subEmail(SnsClient snsClient, String topicArn, String email) {

        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("email")
                .endpoint(email)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n
 Status is " + result.sdkHttpResponse().statusCode());

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

- For API details, see Subscribe in *AWS SDK for Java 2.x API Reference*.

Scenarios

Create a platform endpoint for push notifications

The following code example shows how to create a platform endpoint for Amazon SNS push notifications.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
public class RegistrationExample {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <token>\n\n" +
            "Where:\n" +
            "   token - The name of the FIFO topic. \n\n" +
            "   platformApplicationArn - The ARN value of platform application. You can
 get this value from the AWS Management Console. \n\n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn){

        System.out.println("Creating platform endpoint with token " + token);

        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " + response.endpointArn());
        } catch ( SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Create and publish to a FIFO topic

The following code example shows how to create and publish to a FIFO Amazon SNS topic.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a FIFO topic and FIFO queues. Subscribe the queues to the topic.

```
   public static void main(String[] args) {

       final String usage = "\n" +
           "Usage: " +
           "    <topicArn>\n\n" +
           "Where:\n" +
           "   fifoTopicName - The name of the FIFO topic. \n\n" +
           "   fifoQueueARN - The ARN value of a SQS FIFO queue. You can get this
value from the AWS Management Console. \n\n";

       if (args.length != 2) {
           System.out.println(usage);
           System.exit(1);
       }

       String fifoTopicName = "PriceUpdatesTopic3.fifo";
       String fifoQueueARN = "arn:aws:sqs:us-east-1:814548047983:MyPriceSQS.fifo";
       SnsClient snsClient = SnsClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(ProfileCredentialsProvider.create())
           .build();

       createFIFO(snsClient, fifoTopicName, fifoQueueARN);
   }

   public static void createFIFO(SnsClient snsClient, String topicName, String
queueARN) {

       try {
           // Create a FIFO topic by using the SNS service client.
           Map<String, String> topicAttributes = new HashMap<>();
           topicAttributes.put("FifoTopic", "true");
           topicAttributes.put("ContentBasedDeduplication", "false");

           CreateTopicRequest topicRequest = CreateTopicRequest.builder()
               .name(topicName)
               .attributes(topicAttributes)
               .build();

           CreateTopicResponse response = snsClient.createTopic(topicRequest);
           String topicArn = response.topicArn();
           System.out.println("The topic ARN is"+topicArn);

           // Subscribe to the endpoint by using the SNS service client.
           // Only Amazon SQS FIFO queues can receive notifications from an Amazon SNS
FIFO topic.
           SubscribeRequest subscribeRequest = SubscribeRequest.builder()
               .topicArn(topicArn)
               .endpoint(queueARN)
               .protocol("sqs")
               .build();

           snsClient.subscribe(subscribeRequest);
           System.out.println("The topic is subscribed to the queue.");

           // Compose and publish a message that updates the wholesale price.
           String subject = "Price Update";
           String payload = "{\"product\": 214, \"price\": 79.99}";
           String groupId = "PID-214";
           String dedupId = UUID.randomUUID().toString();
           String attributeName = "business";
           String attributeValue = "wholesale";

           MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
               .dataType("String")
```

```
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            snsClient.publish(pubRequest);
            System.out.println("Message was published to "+topicArn);

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

Publish SMS messages to a topic

The following code example shows how to:

- Create an Amazon SNS topic.
- Subscribe phone numbers to the topic.
- Publish SMS messages to the topic so that all subscribed phone numbers receive the message at once.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a topic and return its ARN.

```
    public static String createSNSTopic(SnsClient snsClient, String topicName ) {

        CreateTopicResponse result = null;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

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

Subscribe an endpoint to a topic.

```
    public static void subTextSNS( SnsClient snsClient, String topicArn, String
 phoneNumber) {

        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("sms")
                .endpoint(phoneNumber)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n
 Status is " + result.sdkHttpResponse().statusCode());

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

Set attributes on the message, such as the ID of the sender, the maximum price, and its type. Message attributes are optional.

```
public class SetSMSAttributes {
    public static void main(String[] args) {

        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket" );

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes( SnsClient snsClient, HashMap<String, String>
 attributes) {

        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
 was " + result.sdkHttpResponse().statusCode());

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

Publish a message to a topic. The message is sent to every subscriber.

```
    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
```

```
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .phoneNumber(phoneNumber)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out.println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

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

Amazon SQS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon SQS.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a queue

The following code example shows how to create an Amazon SQS queue.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String createQueue(SqsClient sqsClient,String queueName ) {

        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");
```

```
            GetQueueUrlResponse getQueueUrlResponse =
sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();

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

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl ) {
        // List queues with filters
        String namePrefix = "queue";
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
            .queueNamePrefix(namePrefix)
            .build();

        ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
        }

        System.out.println("\nSend message");
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello world!")
                .delaySeconds(10)
                .build());

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                .queueUrl(queueUrl)
```

```
.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
               .build();
           sqsClient.sendMessageBatch(sendMessageBatchRequest);

       } catch (SqsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static List<Message> receiveMessages(SqsClient sqsClient, String queueUrl) {

       System.out.println("\nReceive messages");
       try {
           ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
               .queueUrl(queueUrl)
               .maxNumberOfMessages(5)
               .build();
           return sqsClient.receiveMessage(receiveMessageRequest).messages();

       } catch (SqsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
   }

   public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

       System.out.println("\nChange Message Visibility");
       try {

           for (Message message : messages) {
               ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                   .queueUrl(queueUrl)
                   .receiptHandle(message.receiptHandle())
                   .visibilityTimeout(100)
                   .build();
               sqsClient.changeMessageVisibility(req);
           }

       } catch (SqsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
       System.out.println("\nDelete Messages");

       try {
           for (Message message : messages) {
               DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                   .queueUrl(queueUrl)
                   .receiptHandle(message.receiptHandle())
                   .build();
```

```
                sqsClient.deleteMessage(deleteMessageRequest);
            }

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

- For API details, see CreateQueue in *AWS SDK for Java 2.x API Reference*.

Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .receiptHandle(message.receiptHandle())
                    .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
```

- For API details, see DeleteMessage in *AWS SDK for Java 2.x API Reference*.

Delete a queue

The following code example shows how to delete an Amazon SQS queue.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
```

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

- For API details, see DeleteQueue in *AWS SDK for Java 2.x API Reference*.

Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
            GetQueueUrlResponse getQueueUrlResponse =
sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();

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

- For API details, see GetQueueUrl in *AWS SDK for Java 2.x API Reference*.

List queues

The following code example shows how to list Amazon SQS queues.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

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

- For API details, see ListQueues in *AWS SDK for Java 2.x API Reference*.

Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .maxNumberOfMessages(5)
                .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

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

- For API details, see ReceiveMessage in *AWS SDK for Java 2.x API Reference*.

Send a batch of messages to a queue

The following code example shows how to send a batch of messages to an Amazon SQS queue.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
                .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- For API details, see SendMessageBatch in *AWS SDK for Java 2.x API Reference*.

Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void sendMessage(SqsClient sqsClient, String queueName, String
message) {

       try {
           CreateQueueRequest request = CreateQueueRequest.builder()
               .queueName(queueName)
               .build();
           sqsClient.createQueue(request);

           GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
               .queueName(queueName)
               .build();

           String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
           SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
               .queueUrl(queueUrl)
               .messageBody(message)
               .delaySeconds(5)
               .build();

           sqsClient.sendMessage(sendMsgRequest);

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

- For API details, see SendMessage in *AWS SDK for Java 2.x API Reference*.

AWS STS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS STS.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Assume a role

The following code example shows how to assume a role with AWS STS.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {

       try {
           AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
               .roleArn(roleArn)
               .roleSessionName(roleSessionName)
               .build();

          AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
          Credentials myCreds = roleResponse.credentials();

          // Display the time when the temp creds expire.
          Instant exTime = myCreds.expiration();
          String tokenInfo = myCreds.sessionToken();

          // Convert the Instant to readable date.
          DateTimeFormatter formatter =
                  DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                          .withLocale( Locale.US)
                          .withZone( ZoneId.systemDefault() );

          formatter.format( exTime );
          System.out.println("The token "+tokenInfo + "  expires on " + exTime );

      } catch (StsException e) {
          System.err.println(e.getMessage());
          System.exit(1);
      }
  }
```

- For API details, see AssumeRole in *AWS SDK for Java 2.x API Reference*.

Secrets Manager examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Secrets Manager.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Create a secret

The following code example shows how to create a Secrets Manager secret.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createNewSecret( SecretsManagerClient secretsClient, String
secretName, String secretValue) {

       try {
           CreateSecretRequest secretRequest = CreateSecretRequest.builder()
               .name(secretName)
               .description("This secret was created by the AWS Secret Manager Java
API")
               .secretString(secretValue)
               .build();

           CreateSecretResponse secretResponse =
secretsClient.createSecret(secretRequest);
           return secretResponse.arn();

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

- For API details, see CreateSecret in *AWS SDK for Java 2.x API Reference*.

Delete a secret

The following code example shows how to delete a Secrets Manager secret.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void deleteSpecificSecret(SecretsManagerClient secretsClient, String
secretName) {

       try {
           DeleteSecretRequest secretRequest = DeleteSecretRequest.builder()
               .secretId(secretName)
               .build();

           secretsClient.deleteSecret(secretRequest);
           System.out.println(secretName +" is deleted.");

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

- For API details, see DeleteSecret in *AWS SDK for Java 2.x API Reference*.

Describe a secret

The following code example shows how to describe a Secrets Manager secret.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeGivenSecret(SecretsManagerClient secretsClient, String
secretName) {

       try {
           DescribeSecretRequest secretRequest = DescribeSecretRequest.builder()
               .secretId(secretName)
               .build();

           DescribeSecretResponse secretResponse =
secretsClient.describeSecret(secretRequest);
           Instant lastChangedDate = secretResponse.lastChangedDate();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
               DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                       .withLocale( Locale.US)
                       .withZone( ZoneId.systemDefault() );

           formatter.format( lastChangedDate );
           System.out.println("The date of the last change to "+ secretResponse.name()
+" is " + lastChangedDate );

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

- For API details, see DescribeSecret in *AWS SDK for Java 2.x API Reference*.

Get a secret value

The following code example shows how to get a Secrets Manager secret value.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void getValue(SecretsManagerClient secretsClient,String secretName) {

       try {
           GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
               .secretId(secretName)
               .build();

           GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
           String secret = valueResponse.secretString();
           System.out.println(secret);

       } catch (SecretsManagerException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see GetSecretValue in *AWS SDK for Java 2.x API Reference*.

List secrets

The following code example shows how to list Secrets Manager secrets.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllSecrets(SecretsManagerClient secretsClient) {
        try {
            ListSecretsResponse secretsResponse = secretsClient.listSecrets();
            List<SecretListEntry> secrets = secretsResponse.secretList();
            for (SecretListEntry secret: secrets) {
                System.out.println("The secret name is "+secret.name());
                System.out.println("The secret descreiption is "+secret.description());
            }

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

- For API details, see ListSecrets in *AWS SDK for Java 2.x API Reference*.

Put a value in a secret

The following code example shows how to put a value in a Secrets Manager secret.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void updateMySecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {

        try {
            UpdateSecretRequest secretRequest = UpdateSecretRequest.builder()
                .secretId(secretName)
                .secretString(secretValue)
                .build();

            secretsClient.updateSecret(secretRequest);

        } catch (SecretsManagerException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see PutSecretValue in *AWS SDK for Java 2.x API Reference*.

Step Functions examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Step Functions.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello Step Functions

The following code examples show how to get started using Step Functions.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Java version of Hello.

```
    public static void listMachines(SfnClient sfnClient) {

        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine :machines) {
                System.out.println("The name of the state machine is:
"+machine.name());
                System.out.println("The ARN value is : "+machine.stateMachineArn());
            }

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

- For API details, see ListStateMachines in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Create a state machine

The following code example shows how to create a Step Functions state machine.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createMachine( SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
       try {
           CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
               .definition(json)
               .name(stateMachineName)
               .roleArn(roleARN)
               .type(StateMachineType.STANDARD)
               .build();

           CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
           return response.stateMachineArn();

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

- For API details, see CreateStateMachine in *AWS SDK for Java 2.x API Reference*.

Create an activity

The following code example shows how to create a Step Functions activity.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createActivity(SfnClient sfnClient, String activityName) {
       try {
           CreateActivityRequest activityRequest = CreateActivityRequest.builder()
               .name(activityName)
               .build();

           CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
           return response.activityArn();

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

```
        }
        return "";
    }
```

- For API details, see CreateActivity in *AWS SDK for Java 2.x API Reference*.

Delete a state machine

The following code example shows how to delete a Step Functions state machine.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status is
"+response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn +" was successfully deleted.");
    }
```

- For API details, see DeleteStateMachine in *AWS SDK for Java 2.x API Reference*.

Delete an activity

The following code example shows how to delete a Step Functions activity.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
```

```
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted "+actArn);

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

- For API details, see DeleteActivity in *AWS SDK for Java 2.x API Reference*.

Describe a state machine

The following code example shows how to describe a Step Functions state machine.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
       try {
           DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
               .stateMachineArn(stateMachineArn)
               .build();

           DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
           System.out.println("The name of the State machine is "+ response.name());
           System.out.println("The status of the State machine is "+
response.status());
           System.out.println("The ARN value of the State machine is "+
response.stateMachineArn());
           System.out.println("The role ARN value is "+ response.roleArn());

       } catch (SfnException e) {
           System.err.println(e.getMessage());
       }
   }
```

- For API details, see DescribeStateMachine in *AWS SDK for Java 2.x API Reference*.

Describe a state machine run

The following code example shows how to describe a Step Functions state machine run.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeExe(SfnClient sfnClient, String executionArn) {
       try {
           DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
               .executionArn(executionArn)
               .build();

           String status = "";
           boolean hasSucceeded = false;
           while (!hasSucceeded) {
               DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
               status = response.statusAsString();
               if (status.compareTo("RUNNING") ==0) {
                   System.out.println("The state machine is still running, let's wait
for it to finish.");
                   Thread.sleep(2000);
               } else if (status.compareTo("SUCCEEDED") ==0) {
                   System.out.println("The Step Function workflow has succeeded");
                   hasSucceeded = true;
               } else {
                   System.out.println("The Status is neither running or succeeded");
               }
           }
           System.out.println("The Status is "+status);

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

- For API details, see DescribeExecution in *AWS SDK for Java 2.x API Reference*.

Get task data for an activity

The following code example shows how to get task data for a Step Functions activity.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static List<String> getActivityTask(SfnClient sfnClient, String actArn){
       List<String> myList = new ArrayList<>();
       GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
           .activityArn(actArn)
           .build();

       GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
       myList.add(response.taskToken());
       myList.add(response.input());
       return myList;
   }
```

- For API details, see GetActivityTask in *AWS SDK for Java 2.x API Reference*.

List activities

The following code example shows how to list Step Functions activities.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listAllActivites(SfnClient sfnClient) {

        try {
            ListActivitiesRequest activitiesRequest = ListActivitiesRequest.builder()
                .maxResults(10)
                .build();

            ListActivitiesResponse response =
sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item: items) {
                System.out.println("The activity ARN is "+item.activityArn());
                System.out.println("The activity name is "+item.name());
            }

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

- For API details, see ListActivities in *AWS SDK for Java 2.x API Reference*.

List state machine runs

The following code example shows how to list Step Functions state machine runs.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getExeHistory(SfnClient sfnClient, String exeARN) {
        try {
            GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
                .executionArn(exeARN)
                .maxResults(10)
                .build();

            GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
            List<HistoryEvent> events = historyResponse.events();
            for (HistoryEvent event: events) {
                System.out.println("The event type is "+event.type().toString());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

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

- For API details, see ListExecutions in *AWS SDK for Java 2.x API Reference*.

List state machines

The following code example shows how to list Step Functions state machines.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void listMachines(SfnClient sfnClient) {

        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine :machines) {
                System.out.println("The name of the state machine is:
"+machine.name());
                System.out.println("The ARN value is : "+machine.stateMachineArn());
            }

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

- For API details, see ListStateMachines in *AWS SDK for Java 2.x API Reference*.

Send a success response to a task

The following code example shows how to send a success response to a Step Functions task.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void sendTaskSuccess(SfnClient sfnClient, String token, String json)
{
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

           sfnClient.sendTaskSuccess(successRequest);

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

```
        }
    }
```

- For API details, see SendTaskSuccess in *AWS SDK for Java 2.x API Reference*.

Start a state machine run

The following code example shows how to start a Step Functions state machine run.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
       UUID uuid = UUID.randomUUID();
       String uuidValue = uuid.toString();
       try {
           StartExecutionRequest executionRequest = StartExecutionRequest.builder()
               .input(jsonEx)
               .stateMachineArn(stateMachineArn)
               .name(uuidValue)
               .build();

           StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
           return response.executionArn();

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

- For API details, see StartExecution in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with state machines

The following code example shows how to:

- Create an activity.
- Create a state machine from an Amazon States Language definition that contains the previously created activity as a step.
- Run the state machine and respond to the activity with user input.
- Get the final status and output after the run completes, then clean up resources.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * You can obtain the JSON file to create a state machine in the following GitHub
 location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 *
 * To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[]args) throws Exception {
        final String usage = "\n" +
            "Usage:\n" +
            "    <roleARN> <activityName> <stateMachineName>\n\n" +
            "Where:\n" +
            "    roleName - The name of the IAM role to create for this state machine.
\n" +
            "    activityName - The name of an activity to create." +
            "    stateMachineName - The name of the state machine to create.\n";

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String polJSON = "{\n" +
            "    \"Version\": \"2012-10-17\",\n" +
            "    \"Statement\": [\n" +
            "        {\n" +
            "            \"Sid\": \"\",\n" +
            "            \"Effect\": \"Allow\",\n" +
            "            \"Principal\": {\n" +
            "                \"Service\": \"states.amazonaws.com\"\n" +
            "            },\n" +
            "            \"Action\": \"sts:AssumeRole\"\n" +
            "        }\n" +
            "    ]\n" +
            "}" ;

        Scanner sc = new Scanner(System.in);
        boolean action = false ;

        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
```

```
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(regionGl)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Step Functions example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an activity.");
        String activityArn = createActivity(sfnClient, activityName);
        System.out.println("The ARN of the activity is "+activityArn);
        System.out.println(DASHES);

        // Get JSON to use for the state machine and place the activityArn value into
it.
        InputStream input =
StepFunctionsScenario.class.getClassLoader().getResourceAsStream("chat_sfn_state_machine.json");
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
        String jsonString = mapper.writeValueAsString(jsonNode);

        // Modify the Resource node.
        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode root = objectMapper.readTree(jsonString);
        ((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

        // Convert the modified Java object back to a JSON string.
        String stateDefinition = objectMapper.writeValueAsString(root);
        System.out.println(stateDefinition);

        System.out.println(DASHES);
        System.out.println("2. Create a state machine.");
        String roleARN = createIAMRole(iam, roleName, polJSON );
        String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
        System.out.println("The ARN of the state machine is "+stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe the state machine.");
        describeStateMachine(sfnClient, stateMachineArn);
        System.out.println("What should ChatSFN call you?");
        String userName = sc.nextLine();
        System.out.println("Hello "+userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        // The JSON to pass to the StartExecution call.
        String executionJson = "{ \"name\" : \""+userName +"\" }";
        System.out.println(executionJson);
        System.out.println("4. Start execution of the state machine and interact with
it.");
        String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
        System.out.println("The ARN of the state machine execution is "+runArn);
        List<String> myList ;
        while (!action) {
            myList = getActivityTask(sfnClient, activityArn);
            System.out.println("ChatSFN: " + myList.get(1));
```

```
            System.out.println(userName + " please specify a value.");
            String myAction = sc.nextLine();
            if (myAction.compareTo("done") == 0)
                action = true;

            System.out.println("You have selected " + myAction);
            String taskJson = "{ \"action\" : \"" + myAction + "\" }";
            System.out.println(taskJson);
            sendTaskSuccess(sfnClient, myList.get(0), taskJson);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the execution.");
        describeExe(sfnClient, runArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Delete the activity.");
        deleteActivity(sfnClient, activityArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the state machines.");
        deleteMachine(sfnClient, stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is complete.");
        System.out.println(DASHES);
    }

    public static String createIAMRole(IamClient iam, String rolename, String polJSON )
{
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(polJSON)
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

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

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") ==0) {
```

```
                    System.out.println("The state machine is still running, let's wait
for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") ==0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or succeeded");
                }
            }
            System.out.println("The Status is "+status);

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

    public static void sendTaskSuccess(SfnClient sfnClient, String token, String json)
{
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

           sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> getActivityTask(SfnClient sfnClient, String actArn){
        List<String> myList = new ArrayList<>();
        GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
            .activityArn(actArn)
            .build();

        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted "+actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
        try {
```

```
            DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is "+ response.name());
            System.out.println("The status of the State machine is "+
response.status());
            System.out.println("The ARN value of the State machine is "+
response.stateMachineArn());
            System.out.println("The role ARN value is "+ response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status is
"+response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn +" was successfully deleted.");
    }

    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();

            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

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

```
    }

    public static String createMachine( SfnClient sfnClient, String roleARN, String
 stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
 CreateStateMachineRequest.builder()
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
                .build();

            CreateStateMachineResponse response =
 sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

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

    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
 sfnClient.createActivity(activityRequest);
            return response.activityArn();

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - Hello
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution

AWS Support examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Support.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Get started

Hello AWS Support

The following code examples show how to get started using AWS Support.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  In addition, you must have the AWS Business Support Plan to use the AWS Support
 Java API. For more information, see:
 *
 *  https://aws.amazon.com/premiumsupport/plans/
 *
 *  This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
      }

   // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest = DescribeServicesRequest.builder()
                .language("en")
                .build();
```

```
            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service: services) {
                if (index== 11)
                    break;

                System.out.println("The Service name is: "+service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat: categories) {
                    System.out.println("The category name is: "+cat.name());
                }
                index++ ;
            }

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

- For API details, see DescribeServices in *AWS SDK for Java 2.x API Reference*.

Topics

- Actions (p. 224)
- Scenarios (p. 238)

Actions

Add a communication to a case

The following code example shows how to add an AWS Support communication with an attachment to a support case.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void addAttachSupportCase(SupportClient supportClient, String caseId,
 String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
 AddCommunicationToCaseRequest.builder()
                .caseId(caseId)
                .attachmentSetId(attachmentSetId)
                .communicationBody("Please refer to attachment for details.")
                .build();

            AddCommunicationToCaseResponse response =
 supportClient.addCommunicationToCase(caseRequest);
```

```
            if (response.result())
                System.out.println("You have successfully added a communication to an
AWS Support case");
            else
                System.out.println("There was an error adding the communication to an
AWS Support case");

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

- For API details, see AddCommunicationToCase in *AWS SDK for Java 2.x API Reference*.

Add an attachment to a set

The following code example shows how to add an AWS Support attachment to an attachment set.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
                .fileName(myFile.getName())
                .data(sourceBytes)
                .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                .attachments(attachment)
                .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

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

- For API details, see AddAttachmentsToSet in *AWS SDK for Java 2.x API Reference*.

Create a case

The following code example shows how to create a new AWS Support case.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
       try {
           String serviceCode = sevCatList.get(0);
           String caseCat = sevCatList.get(1);
           CreateCaseRequest caseRequest = CreateCaseRequest.builder()
               .categoryCode(caseCat.toLowerCase())
               .serviceCode(serviceCode.toLowerCase())
               .severityCode(sevLevel.toLowerCase())
               .communicationBody("Test issue with "+serviceCode.toLowerCase())
               .subject("Test case, please ignore")
               .language("en")
               .issueType("technical")
               .build();

           CreateCaseResponse response = supportClient.createCase(caseRequest);
           return response.caseId();

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

- For API details, see CreateCase in *AWS SDK for Java 2.x API Reference*.

Describe an attachment

The following code example shows how to describe an attachment for an AWS Support case.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static void describeAttachment(SupportClient supportClient,String attachId)
{
       try {
           DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
               .attachmentId(attachId)
               .build();

           DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
           System.out.println("The name of the file is
"+response.attachment().fileName());

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

```
    }
```

- For API details, see DescribeAttachment in *AWS SDK for Java 2.x API Reference*.

Describe cases

The following code example shows how to describe AWS Support cases.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest = DescribeCasesRequest.builder()
                .maxResults(20)
                .afterTime(yesterday.toString())
                .beforeTime(now.toString())
                .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase: cases) {
                System.out.println("The case status is "+sinCase.status());
                System.out.println("The case Id is "+sinCase.caseId());
                System.out.println("The case subject is "+sinCase.subject());
            }

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

- For API details, see DescribeCases in *AWS SDK for Java 2.x API Reference*.

Describe communications

The following code example shows how to describe AWS Support communications for a case.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String listCommunications(SupportClient supportClient, String caseId)
{
        try {
```

```
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                .caseId(caseId)
                .maxResults(10)
                .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm: communications) {
                System.out.println("the body is: " + comm.body());

                //Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

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

- For API details, see DescribeCommunications in *AWS SDK for Java 2.x API Reference*.

Describe services

The following code example shows how to describe the list of AWS services.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest = DescribeServicesRequest.builder()
                .language("en")
                .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service: services) {
                if (index== 11)
                    break;

                System.out.println("The Service name is: "+service.name());
```

```
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat: categories) {
                    System.out.println("The category name is: "+cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
             index++ ;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

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

- For API details, see DescribeServices in *AWS SDK for Java 2.x API Reference*.

Describe severity levels

The following code example shows how to describe AWS Support severity levels.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                .language("en")
                .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel: severityLevels) {
                System.out.println("The severity level name is: "+ sevLevel.name());
                if (sevLevel.name().compareTo("High")==0)
                    levelName = sevLevel.name();
            }
            return levelName;

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

- For API details, see DescribeSeverityLevels in *AWS SDK for Java 2.x API Reference*.

Resolve case

The following code example shows how to resolve an AWS Support case.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void resolveSupportCase(SupportClient supportClient, String caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                .caseId(caseId)
                .build();

            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case "+caseId +" is
 "+response.finalCaseStatus());

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

- For API details, see ResolveCase in *AWS SDK for Java 2.x API Reference*.

Scenarios

Get started with cases

The following code example shows how to:

- Get and display available services and severity levels for cases.
- Create a support case using a selected service, category, and severity level.
- Get and display a list of open cases for the current day.
- Add an attachment set and a communication to the new case.
- Describe the new attachment and communication for the case.
- Resolve the case.
- Get and display a list of resolved cases for the current day.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Run various AWS Support operations.

```
/**
```

```
 * Before running this Java (v2) code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  In addition, you must have the AWS Business Support Plan to use the AWS Support
 Java API. For more information, see:
 *
 *  https://aws.amazon.com/premiumsupport/plans/
 *
 *  This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and severity
 level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fileAttachment>" +
            "Where:\n" +
            "    fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileAttachment = args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
 category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
```

```
        if (caseId.compareTo("")==0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case "+caseId +" was successfully created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add to
the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" +attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" +attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest = DescribeCasesRequest.builder()
                .maxResults(30)
                .afterTime(yesterday.toString())
                .beforeTime(now.toString())
                .includeResolvedCases(true)
                .build();
```

```
            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase: cases) {
                if (sinCase.status().compareTo("resolved") ==0)
                    System.out.println("The case status is "+sinCase.status());
            }

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

    public static void resolveSupportCase(SupportClient supportClient, String caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                .caseId(caseId)
                .build();

            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case "+caseId +" is
"+response.finalCaseStatus());

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

    public static void describeAttachment(SupportClient supportClient,String attachId)
{
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                .attachmentId(attachId)
                .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is
"+response.attachment().fileName());

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

    public static String listCommunications(SupportClient supportClient, String caseId)
{
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                .caseId(caseId)
                .maxResults(10)
                .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm: communications) {
                System.out.println("the body is: " + comm.body());

                //Get the attachment id value.
```

```
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

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

    public static void addAttachSupportCase(SupportClient supportClient, String caseId,
String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                .caseId(caseId)
                .attachmentSetId(attachmentSetId)
                .communicationBody("Please refer to attachment for details.")
                .build();

            AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication to an
AWS Support case");
            else
                System.out.println("There was an error adding the communication to an
AWS Support case");

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

    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
                .fileName(myFile.getName())
                .data(sourceBytes)
                .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                .attachments(attachment)
                .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

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

```
   public static void getOpenCase(SupportClient supportClient) {
       try {
           // Specify the start and end time.
           Instant now = Instant.now();
           java.time.LocalDate.now();
           Instant yesterday = now.minus(1, ChronoUnit.DAYS);

           DescribeCasesRequest describeCasesRequest = DescribeCasesRequest.builder()
               .maxResults(20)
               .afterTime(yesterday.toString())
               .beforeTime(now.toString())
               .build();

           DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
           List<CaseDetails> cases = response.cases();
           for (CaseDetails sinCase: cases) {
               System.out.println("The case status is "+sinCase.status());
               System.out.println("The case Id is "+sinCase.caseId());
               System.out.println("The case subject is "+sinCase.subject());
           }

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

   public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
       try {
           String serviceCode = sevCatList.get(0);
           String caseCat = sevCatList.get(1);
           CreateCaseRequest caseRequest = CreateCaseRequest.builder()
               .categoryCode(caseCat.toLowerCase())
               .serviceCode(serviceCode.toLowerCase())
               .severityCode(sevLevel.toLowerCase())
               .communicationBody("Test issue with "+serviceCode.toLowerCase())
               .subject("Test case, please ignore")
               .language("en")
               .issueType("technical")
               .build();

           CreateCaseResponse response = supportClient.createCase(caseRequest);
           return response.caseId();

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

   public static String displaySevLevels(SupportClient supportClient) {
       try {
           DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
               .language("en")
               .build();

           DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
           List<SeverityLevel> severityLevels = response.severityLevels();
           String levelName = null;
           for (SeverityLevel sevLevel: severityLevels) {
```

```
                System.out.println("The severity level name is: "+ sevLevel.name());
                if (sevLevel.name().compareTo("High")==0)
                    levelName = sevLevel.name();
            }
            return levelName;

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

    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest = DescribeServicesRequest.builder()
                .language("en")
                .build();

            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service: services) {
                if (index== 11)
                    break;

                System.out.println("The Service name is: "+service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat: categories) {
                    System.out.println("The category name is: "+cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
             index++ ;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

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

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AddAttachmentsToSet
  - AddCommunicationToCase

- CreateCase
- DescribeAttachment
- DescribeCases
- DescribeCommunications
- DescribeServices
- DescribeSeverityLevels
- ResolveCase

Amazon Textract examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Textract.

Actions are code excerpts that show you how to call individual service functions.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- Actions (p. 224)

Actions

Analyze a document

The following code example shows how to analyze a document using Amazon Textract.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {

        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            List<FeatureType> featureTypes = new ArrayList<FeatureType>();
            featureTypes.add(FeatureType.FORMS);
            featureTypes.add(FeatureType.TABLES);

            AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
                    .featureTypes(featureTypes)
                    .document(myDoc)
                    .build();
```

```
            AnalyzeDocumentResponse analyzeDocument =
textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();

            while(blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is " +block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see AnalyzeDocument in *AWS SDK for Java 2.x API Reference*.

Detect text in a document

The following code example shows how to detect text in a document using Amazon Textract.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Detect text from an input document.

```
    public static void detectDocText(TextractClient textractClient,String sourceDoc) {

        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                .bytes(sourceBytes)
                .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                .document(myDoc)
                .build();

            // Invoke the Detect operation
            DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is "
+documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
```

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

Detect text from a document located in an Amazon S3 bucket.

```
   public static void detectDocTextS3 (TextractClient textractClient, String
bucketName, String docName) {

       try {
           S3Object s3Object = S3Object.builder()
               .bucket(bucketName)
               .name(docName)
               .build();

           // Create a Document object and reference the s3Object instance
           Document myDoc = Document.builder()
               .s3Object(s3Object)
               .build();

           DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
               .document(myDoc)
               .build();

           DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
           for (Block block: textResponse.blocks()) {
               System.out.println("The block type is " +block.blockType().toString());
           }

           DocumentMetadata documentMetadata = textResponse.documentMetadata();
           System.out.println("The number of pages in the document is "
+documentMetadata.pages());

       } catch (TextractException e) {

           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- For API details, see DetectDocumentText in *AWS SDK for Java 2.x API Reference*.

Start asynchronous analysis of a document

The following code example shows how to start asynchronous analysis of a document using Amazon Textract.

SDK for Java 2.x

> **Note**
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   public static String startDocAnalysisS3 (TextractClient textractClient, String
bucketName, String docName) {

       try {
           List<FeatureType> myList = new ArrayList<>();
```

```
            myList.add(FeatureType.TABLES);
            myList.add(FeatureType.FORMS);

            S3Object s3Object = S3Object.builder()
                .bucket(bucketName)
                .name(docName)
                .build();

            DocumentLocation location = DocumentLocation.builder()
                .s3Object(s3Object)
                .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                .documentLocation(location)
                .featureTypes(myList)
                .build();

            StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

            // Get the job ID
            String jobId = response.jobId();
            return jobId;

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

    private static String getJobResults(TextractClient textractClient, String jobId) {

        boolean finished = false;
        int index = 0 ;
        String status = "" ;

       try {
           while (!finished) {
               GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                   .jobId(jobId)
                   .maxResults(1000)
                   .build();

               GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
               status = response.jobStatus().toString();

               if (status.compareTo("SUCCEEDED") == 0)
                   finished = true;
               else {
                   System.out.println(index + " status is: " + status);
                   Thread.sleep(1000);
               }
               index++ ;
           }

           return status;

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

```
    }
```

- For API details, see StartDocumentAnalysis in *AWS SDK for Java 2.x API Reference*.

Cross-service examples using SDK for Java 2.x

The following sample applications use the AWS SDK for Java 2.x to work across multiple AWS services.

Examples

- Build an application to submit data to a DynamoDB table (p. 656)
- Create an Amazon Lex chatbot to engage your website visitors (p. 656)
- Build a publish and subscription application that translates messages (p. 657)
- Create a web application that sends and retrieves messages by using Amazon SQS (p. 657)
- Create a web application to track DynamoDB data (p. 657)
- Create an Amazon Redshift item tracker (p. 657)
- Create an Aurora Serverless work item tracker (p. 658)
- Detect PPE in images with Amazon Rekognition using an AWS SDK (p. 658)
- Detect objects in images with Amazon Rekognition using an AWS SDK (p. 658)
- Detect people and objects in a video with Amazon Rekognition using an AWS SDK (p. 659)
- Use API Gateway to invoke a Lambda function (p. 659)
- Use Step Functions to invoke Lambda functions (p. 659)
- Use scheduled events to invoke a Lambda function (p. 660)

Build an application to submit data to a DynamoDB table

SDK for Java 2.x

Shows how to create a dynamic web application that submits data using the Amazon DynamoDB Java API and sends a text message using the Amazon Simple Notification Service Java API.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- Amazon SNS

Create an Amazon Lex chatbot to engage your website visitors

SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Build a publish and subscription application that translates messages

SDK for Java 2.x

Shows how to use the Amazon Simple Notification Service Java API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

For complete source code and instructions on how to set up and run the example that uses the Java Async API, see the full example on GitHub.

Services used in this example

- Amazon SNS
- Amazon Translate

Create a web application that sends and retrieves messages by using Amazon SQS

SDK for Java 2.x

Shows how to use the Amazon SQS API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon SQS

Create a web application to track DynamoDB data

SDK for Java 2.x

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- Amazon SES

Create an Amazon Redshift item tracker

SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on GitHub.

Services used in this example

- Amazon Redshift
- Amazon SES

Create an Aurora Serverless work item tracker

SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on GitHub.

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on GitHub.

Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Detect PPE in images with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detect objects in images with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3)

bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detect people and objects in a video with Amazon Rekognition using an AWS SDK

SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Use API Gateway to invoke a Lambda function

SDK for Java 2.x

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

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Use Step Functions to invoke Lambda functions

SDK for Java 2.x

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Use scheduled events to invoke a Lambda function

SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Security for the AWS SDK for Java

Cloud security at Amazon Web Services (AWS) is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The Shared Responsibility Model describes this as Security of the Cloud and Security in the Cloud.

Security of the Cloud – AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at AWS, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS Compliance Programs.

Security in the Cloud – Your responsibility is determined by the AWS service you are using, and other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Topics

- Data protection in AWS SDK for Java 2.x (p. 661)
- Working with TLS in the SDK for Java (p. 662)
- Identity and Access Management (p. 663)
- Compliance Validation for this AWS Product or Service (p. 666)
- Resilience for this AWS Product or Service (p. 666)
- Infrastructure Security for this AWS Product or Service (p. 667)

Data protection in AWS SDK for Java 2.x

The AWS shared responsibility model applies to data protection in AWS SDK for Java. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center (successor to AWS Single Sign-On) or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.

- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with SDK for Java or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Working with TLS in the SDK for Java

The AWS SDK for Java uses the TLS capabilities of its underlying Java platform. In this topic, we show examples using the OpenJDK implementation used by Amazon Corretto 18.

Users should consult the documentation of the the Java platform they are using with the SDK to find out which TLS versions are enabled by default as well as how to enable and disable specific TLS versions.

How to check TLS version information

Using OpenJDK, the following code shows the use of SSLContext to print which TLS/SSL versions are supported.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()))
```

For example, Amazon Corretto 18 (OpenJDK) produces the following output.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

To see the SSL handshake in action and what version of TLS is used, you can use the system property **javax.net.debug**.

For example, run a Java applications that uses TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

The application logs the SSL handshake similar to the following.

```
...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.221 EST|ClientHello.java:641|Produced
 ClientHello handshake message (
"ClientHello": {
  "client version"      : "TLSv1.2",

...
javax.net.ssl|DEBUG|10|main|2022-12-23 13:53:12.295 EST|ServerHello.java:888|Consuming
 ServerHello handshake message (
"ServerHello": {
  "server version"      : "TLSv1.2",
```

```
...
```

Enforce a minimum TLS version

The SDK for Java always prefers the latest TLS version supported by the platform and service. If you wish to enforce a specific minimum TLS version, consult your Java platform's documentation.

For OpenJDK-based JVMs, you can use the system property `jdk.tls.client.protocols`.

For example, if you want SDK service clients in your application to use TLS 1.2, even though TLS 1.3 is available, provide the following system property.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWS API endpoints upgrade to TLS 1.2

See this blog post for information about AWS API endpoints moving to TLS 1.2 for the minimum version.

AWS Open Source Blog has related information about continued support for TLS 1.0 and TLS 1.1 with OpenJDK and Amazon Corretto.

Identity and Access Management

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS resources. IAM is an AWS service that you can use with no additional charge.

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS.

Service user – If you use AWS services to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator.

Service administrator – If you're in charge of AWS resources at your company, you probably have full access to AWS resources. It's your job to determine which AWS services and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS services.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (successor to AWS Single Sign-On) (IAM Identity Center) users, your company's

single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

To access AWS programmatically, AWS provides software development kits (SDKs) and a command line interface (CLI) to cryptographically sign your requests using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *AWS Account Management Reference Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center (successor to AWS Single Sign-On). You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to

manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Compliance Validation for this AWS Product or Service

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

The security and compliance of AWS services is assessed by third-party auditors as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others. AWS provides a frequently updated list of AWS services in scope of specific compliance programs at AWS Services in Scope by Compliance Program.

Third-party audit reports are available for you to download using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

For more information about AWS compliance programs, see AWS Compliance Programs.

Your compliance responsibility when using this AWS product or service to access an AWS service is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of an AWS service is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and Compliance Quick Start Guides – Deployment guides that discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – A whitepaper that describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – A collection of workbooks and guides that might apply to your industry and location.
- AWS Config – A service that assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – A comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience for this AWS Product or Service

The Amazon Web Services (AWS) global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Infrastructure Security for this AWS Product or Service

This AWS product or service follows the shared responsibility model through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the AWS service security documentation page and AWS services that are in scope of AWS compliance efforts by compliance program.

Document history

This topic describes important changes to the AWS SDK for Java Developer Guide over the course of its history.

This documentation was last built on: 2023-03-14

Change	Description	Date
IAM best practices updates	Updated guide to align with the IAM best practices. For more information, see Security best practices in IAM.	14 March 2023
the section called “Reload profile credentials” (p. 39)	Add section on reloading profile credentials	9 February 2023
the section called “Configuring the AWS CRT-based HTTP client” (p. 62)	Update topic for GA release	8 February 2023
the section called “Work with Amazon EC2 instance metadata” (p. 156)	Add guided example for Java SDK client for Amazon S3 instance metadata service	1 February 2023
the section called “AWS CRT-based S3 client” (p. 94)	Add section for the AWS CRT-based S3 Client	19 December 2022
the section called “S3 Transfer Manager” (p. 95)	Update Amazon S3 Transfer Manager examples for GA release.	19 December 2022
the section called “Best practices” (p. 74)	Added best practices section	18 November 2022
the section called “Load temporary credentials from an external process” (p. 40)	Added section on loading credentials from an external process	15 November 2022
the section called “Service client metrics” (p. 85)	Updated metric listing with HTTP client usage requirement	9 November 2022
the section called “Mapping items in DynamoDB tables” (p. 131)	Added details for dependencies and using a builder to create a `TableSchema`	4 November 2022
the section called “S3 Transfer Manager” (p. 95)	Example code corrected	2 November 2022
the section called “Reducing SDK startup time for AWS Lambda” (p. 47)	Updated section with additional options to reduce Lambda starup time	1 November 2022
the section called “Configuring HTTP clients” (p. 50)	Added configuration information to cover all HTTP clients in the SDK	26 October 2022

Change	Description	Date
the section called “Mapping items in DynamoDB tables” (p. 131)	Corrected examples	10 October 2022
the section called “Logging” (p. 68)	Updated logging topic to include wire logging details for all HTTP clients	4 October 2022
the section called “AWS database services” (p. 105)	Added overview section of AWS database services and the SDK for Java 2.x	13 September 2022
EC2-Classic Networking is Retiring	EC2-Classic is retiring on August 15, 2022	28 July 2022
the section called “Additional authentication options” (p. 14)	Update to dependency required for single sign-on authentication	18 July 2022
the section called “Transport Layer Security (TLS)” (p. 662)	Update TLS security information	8 April 2022
the section called “Additional authentication options” (p. 14)	Added more information about setting up and using credentials	22 February 2021
the section called “Set up a GraalVM Native Image project” (p. 19)	New topic for setting up a GraalVM Native Image project	18 February 2021
the section called “Waiters” (p. 101)	Waiters released; added topic for the new feature	30 September 2020
the section called “SDK Metrics” (p. 83)	Metrics released; added topic for the new feature	17 August 2020
the section called “ Amazon Pinpoint ” (p. 201), the section called “ Amazon Cognito ” (p. 187), the section called “ Amazon Simple Notification Service ” (p. 209)	Added example topics for Amazon Pinpoint, Amazon Cognito, and Amazon SNS	30 May 2020
the section called “Reducing SDK startup time for AWS Lambda” (p. 47)	Added AWS Lambda function performance topic	29 May 2020
the section called “Setting the JVM TTL for DNS name lookups” (p. 73)	Added JVM TTL DNS caching topic	27 April 2020
the section called “Set up an Apache Maven project” (p. 14), the section called “Set up a Gradle project” (p. 18)	New Maven and Gradle set up topics	21 April 2020
the section called “Mapping items in DynamoDB tables” (p. 131)	Added DynamoDB enhanced client topic	20 April 2020

Change	Description	Date
the section called "Transport Layer Security (TLS)" (p. 662)	Added TLS 1.2 to security section	19 March 2020
the section called "Subscribing to Amazon Kinesis Data Streams" (p. 192)	Added Kinesis stream examples	2 August 2018
the section called "Pagination" (p. 88)	Added auto pagination topic	5 April 2018
Work with AWS services (p. 104)	Added example topics for IAM, Amazon EC2, CloudWatch and DynamoDB	29 December 2017
the section called " Amazon Simple Storage Service (S3)" (p. 106)	Added getobjects example for Amazon S3	7 August 2017
the section called "Asynchronous programming" (p. 77)	Added async topic	4 August 2017
GA release of the AWS SDK for Java 2.x	AWS SDK for Java version 2 (v2) released	28 June 2017