

AWS SDK for Java 2.x

<-·>
↳ — [-] ⠄ { ~w } [...]
[~w] ⠄ <-·> [...] ⠄ { ... } ⠄ [~w] ⠄ <-·>
{ ... } [·—] ~w <-·> ... [-·] ⠄ { ... } [·—] ~w <-·>
·— [~w] [...] <-·> ⠄ — { ... } ⠄ [~w] [...] <-·>
↳ — [-] ⠄ { ~w } [...] ... <-·> ⠄ — [-] ⠄ { ~w } [...]
<-·> ⠄ — { ~w } [...] ⠄ [-] ... <-·> ⠄ — { ~w } [...]
↳ — [-] ⠄ { ~w } [...] ... <-·> ⠄ — [-] ⠄ { ~w } [...]
 <-·> [...] ⠄ { ... } ⠄ [~w] ⠄ <-·> [...]
 ↑ ⠄ { ... } [·—] ~w <-·>

AWS SDK for Java 2.x: Guia do desenvolvedor para a versão 2.x

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigue a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

Guia do desenvolvedor - AWS SDK for Java 2.x	1
Comece a usar o SDK	1
Desenvolva aplicativos móveis	1
Manutenção e suporte para as versões principais do SDK	1
Recursos adicionais	2
Contribua com o SDK	2
Tutorial de conceitos básicos	3
Etapa 1: Configurar este tutorial	3
Etapa 2: criar um projeto	3
Etapa 3: escrever o código	8
Etapa 4: Compilar e executar o aplicativo	12
Bem-sucedida	13
Limpeza	13
Próximas etapas	14
Configuração	15
Configuração básica	16
Visão geral	16
Capacidade de login no portal de acesso AWS	17
Configurar o acesso de login único para o SDK	17
Faça login usando o AWS CLI	18
Instale o Java e uma ferramenta de construção	19
Opcões adicionais de autenticação	19
Configuração de ferramentas de construção	19
Configurar um projeto Apache Maven	19
Configurar um projeto Gradle	25
Configurar um projeto GraalVM Native Image	32
Pré-requisitos	32
Crie um projeto usando o arquétipo	33
Crie uma imagem nativa	33
Use o SDK	35
Trabalhe com clientes de serviços	35
Crie um cliente de serviço	35
Configuração padrão do cliente	36
Configurar clientes de serviço	36

Faça solicitações	37
Lidar com respostas	37
Feche o cliente de serviço	37
Lidar com exceções	38
Use garçons	39
Cientes HTTP	40
Tentativas	40
Tempo limite atingido	40
Interceptores de execução	40
Informações adicionais	40
Forneça credenciais temporárias ao SDK	41
Usar credenciais temporárias	41
Cadeia de provedores de credenciais padrão	43
Use um provedor de credenciais específico ou uma cadeia de fornecedores	45
Use perfis	45
Carregar credenciais temporárias de um processo externo	48
Forneça credenciais temporárias em código	52
Configurar IAMfunções paraAmazon EC2	55
AWSseleção da região	57
Escolher uma região	57
Escolher um endpoint específico	58
Determine automaticamente a região a partir do ambiente	58
Verificando a disponibilidade do serviço em uma região	59
Reduza o tempo de inicialização do SDK paraAWS Lambda	60
Use os SDKsURLConnectionHttpClient	60
Use os SDKsAwsCrtAsyncHttpClient	61
Remover dependências não utilizadas do cliente HTTP	61
Configurar clientes de serviço para pesquisar atalhos	63
Inicialize o cliente SDK fora do manipulador de funções do Lambda	64
Minimize a injeção de dependência	64
Use uma mira do Maven ArchetypeAWS Lambda	64
LambdaSnapStart	65
Alterações na versão 2.x que afetam o tempo de inicialização	65
Recursos adicionais	65
Clientes HTTP	65
Clientes disponíveis	66

Recomendações do cliente	66
Padrões inteligentes	70
Suporte de proxy	72
Configurar o cliente HTTP baseado em Apache	74
Configurar o cliente HTTP baseado em URLConnection	79
Configurar o cliente HTTP baseado em Netty	85
Configurar o cliente AWS HTTP baseado em CRT	91
Tratamento de exceções	98
Por que exceções não verificadas?	98
AwsServiceException (e subclasses)	99
SdkClientException	100
Exceções e comportamento de repetição	100
Registro em log	100
Arquivo de configuração do Log4j 2	100
Adicionar dependência de registro	101
Erros e avisos específicos do SDK	102
Registro resumido da solicitação/resposta	102
Registro detalhado de cabos	103
Definir o TTL da JVM para pesquisas de nomes DNS	109
Como configurar o TTL da JVM	109
Práticas recomendadas	110
Reutilize um cliente SDK	110
Fechar fluxos de entrada	110
Ajustar as configurações HTTP	111
Use o OpenSSL para Netty	111
Configurar tempos limite de API	111
Usar métricas	113
Recursos do SDK	114
Programação assíncrona	114
Operações sem streaming	114
Operações de streaming	117
Operações avançadas	121
API de cliente aprimorada do DynamoDB	123
Conceitos básicos	123
Conceitos básicos	132
Extensões	187

Recursos avançados de esquema de tabela	191
API de documentos aprimorada	224
Operações assíncronas sem bloqueio	237
Anotações	238
HTTP/2	241
Métricas do SDK	241
Pré-requisitos	241
Como ativar a coleta de métricas	242
Quais informações são coletadas?	243
Como posso usar essas informações?	244
Métricas do cliente de serviço	244
Paginação	249
Paginação síncrona	249
Paginação assíncrona	252
AWSCliente S3 baseado em CRT	257
Adicionar dependências	258
Criar uma instância do	258
Use o cliente AWS S3 baseado em CRT	259
Gerenciador de transferências S3	260
Conceitos básicos	260
Fazer upload de um arquivo	262
Faça o download de um arquivo	263
Copiar um objeto	264
Fazer upload de um diretório	265
Faça o download para um diretório	266
Waiters	268
Pré-requisitos	268
Usando garçons	268
Configurar garçons	269
Exemplos de código	270
API IAM Policy Builder	270
Criar um IamPolicy	271
Use e IamPolicy com IAM	276
Trabalhar com Serviços da AWS	279
CloudWatch	279
Obtenha métricas de CloudWatch	280

Publique dados métricos personalizados em CloudWatch	282
Trabalhe com CloudWatch alarmes	284
Use ações de alarme em CloudWatch	288
Enviar eventos para CloudWatch	290
Amazon Cognito	294
Criar um grupo de usuários	294
Listar usuários de um grupo de usuários	295
Criar um grupo de identidades do	296
Adicionar um cliente de aplicativo	297
Adicionar um provedor de identidade de terceiros	298
Obter credenciais para um ID	300
AWSserviços de banco de dados	301
Amazon DynamoDB	301
Amazon RDS	302
Amazon Redshift	302
Amazon Aurora less v1	303
Amazon DocumentDB	303
DynamoDB	303
Trabalhe com tabelas em DynamoDB	304
Trabalhe com itens em DynamoDB	314
Amazon EC2	321
Gerenciar Amazon EC2 instâncias	321
Use endereços IP elásticos em Amazon EC2	328
Zonas Regiões da AWS de uso e disponibilidade	332
Trabalhar comAmazon EC2Pares de chaves do	335
Trabalhar com grupos de segurança noAmazon EC2	338
Trabalhe com metadados da instância do Amazon EC2	342
IAM	349
Gerenciar chaves de IAM acesso	349
Gerenciar IAM usuários	355
Use aliases de IAM conta	360
Trabalhe com IAM políticas	363
Trabalhe com certificados IAM de servidor	369
Kinesis	374
Inscrever-se no Amazon Kinesis Data Streams	374
Lambda	384

Invocar uma função do Lambda	384
Listar funções do Lambda	385
Excluir uma função do Lambda	386
Amazon Pinpoint	387
Criar um projeto	387
Criar um segmento dinâmico	388
Importar um segmento estático	391
Listar segmentos para o projeto	392
Criar uma campanha	394
Enviar uma mensagem	396
Amazon S3	398
Use pontos de acesso ou pontos de acesso multirregionais	399
Operações de buckets	400
Operações de objetos	406
Pre-signed URLs	413
Acesso entre regiões	420
Somas de verificação	421
Amazon SNS	423
Criar um tópico	423
Liste seus Amazon SNS tópicos	424
Inscriver um endpoint em um tópico	425
Publicar uma mensagem em um tópico	426
Cancelar a inscrição de um endpoint de um tópico	427
Excluir um tópico	428
Amazon SQS	429
Operações de fila	429
Operações de mensagens	433
Amazon Transcribe	436
Configurar o microfone	436
Crie um editor	437
Crie o cliente e inicie a transmissão	440
Mais informações	436
Exemplos de código	442
Ações e cenários	442
API Gateway	444
Controlador de recuperação de aplicativos	448

Aurora	451
Auto Scaling	485
CloudFront	542
CloudWatch	558
CloudWatch Eventos	604
CloudWatch Registros	607
Provedor de identidade do Amazon Cognito	611
Amazon Comprehend	629
DynamoDB	636
Amazon EC2	679
Amazon ECS	749
Elastic Load Balancing	756
OpenSearch Serviço	801
EventBridge	806
Previsão	837
AWS Glue	845
HealthImaging	862
IAM	888
Amazon Keyspaces	956
Kinesis	983
AWS KMS	990
Lambda	1000
MediaConvert	1016
Migration Hub	1031
Amazon Personalize	1038
Eventos do Amazon Personalize	1069
Tempo de execução do Amazon Personalize	1072
Amazon Pinpoint	1077
API SMS and Voice do Amazon Pinpoint	1100
Amazon Polly	1102
Amazon RDS	1106
Amazon Redshift	1139
Amazon Rekognition	1144
Registro de domínio do Route 53	1184
Amazon S3	1207
S3 Glacier	1272

SageMaker	1282
Secrets Manager	1311
Amazon SES	1318
API do Amazon SES v2	1325
Amazon SNS	1327
Amazon SQS	1355
Step Functions	1372
AWS STS	1394
AWS Support	1396
Systems Manager	1418
Amazon Textract	1423
Exemplos entre serviços	1429
Criar uma aplicação para enviar dados para uma tabela do DynamoDB	1430
Criando um chatbot Amazon Lex	1430
Criação de uma aplicação do Amazon SNS	1431
Crie um aplicativo de mensagens	1431
Criar uma aplicação com tecnologia sem servidor para gerenciar fotos	1432
Criar uma aplicação Web para monitorar dados do DynamoDB	1432
Criar uma aplicação Web para rastrear dados do Amazon Redshift	1433
Crie um rastreador de itens de trabalho do Aurora Sem Servidor	1433
Criar uma aplicação para analisar o feedback dos clientes	1434
Detectar EPI em imagens	1434
Detectar objetos em imagens	1435
Detectar pessoas e objetos em um vídeo	1435
Publicar mensagens em filas	1436
Usar o API Gateway para invocar uma função do Lambda	1436
Usar Step Functions para invocar funções do Lambda	1437
Usar eventos programados para chamar uma função do Lambda	1437
Segurança	1439
Proteção de dados	1439
Transport Layer Security (TLS)	1441
Verifique as versões do TLS	1441
Imponha as versões do TLS	1442
Migrar para o TLS 1.2	1442
Gerenciamento de identidade e acesso	1442
Público	1443

Como autenticar com identidades	1443
Gerenciamento do acesso usando políticas	1447
Como Serviços da AWS funcionam com o IAM	1450
Solução de problemas de identidade e acesso do AWS	1450
Validação de compatibilidade	1452
Resiliência	1453
Segurança da infraestrutura	1454
Migrar para a versão 2	1455
O que há de novo	1455
O que há de diferente entre 1.x e 2.x	1456
Alteração do nome do pacote	1456
Adicionando a versão 2.x ao seu projeto	1457
Criadores de clientes	1457
Configuração do cliente	1458
Métodos setter	1459
Nomes de classes	1459
Classe de região	1460
POJOs imutáveis	1460
Operações de streaming	1461
Alterações na exceção	1462
Mudanças específicas do serviço	1462
Alterações adicionais do cliente	1463
Alterações no provedor de credenciais	1464
Como renomear um nome de classe	1466
Alterações nos nomes de classe de	1468
Bibliotecas e utilitários	1469
Use o SDK for Java 1.x side-by-side	1471
Histórico do documento	1473

mcdlxix

Guia do desenvolvedor - AWS SDK for Java 2.x

O AWS SDK for Java fornece uma API do Java para o Serviços da AWS. Usando o SDK, você pode criar aplicativos Java que funcionam com Amazon S3, Amazon EC2, Amazon DynamoDB, e muito mais.

O AWS SDK for Java 2.x é uma grande reescrita da base de código da versão 1.x. Ele foi criado com base no Java 8+ e adiciona vários recursos frequentemente solicitados. Isso inclui suporte para E/S sem bloqueio e a capacidade de conectar uma implementação HTTP diferente em tempo de execução.

Sempre adicionamos suporte para novos serviços ao AWS SDK for Java. Para obter uma lista de alterações e recursos em uma versão específica, visualize o [log de alterações](#).

Comece a usar o SDK

Se você estiver pronto para começar a usar o SDK, siga o tutorial. [Tutorial de conceitos básicos](#)

Para configurar seu ambiente de desenvolvimento, consulte [Configuração](#).

Se você estiver usando a versão 1.x do SDK for Java, consulte [Migrar para a versão 2 para obter orientação](#) específica.

Para obter informações sobre como fazer solicitações para Amazon S3, Amazon DynamoDB, Amazon EC2 e outros serviços da AWS, consulte [Usar SDK for Java](#) e [trabalhar com Serviços da AWS](#).

Desenvolva aplicativos móveis

Se você é um desenvolvedor de aplicativos móveis, Amazon Web Services fornece a [AWS Amplify](#) estrutura. Para obter mais informações, consulte a [documentação da Amplify](#).

Manutenção e suporte para as versões principais do SDK

Para obter informações sobre manutenção e suporte para as versões principais do SDK e suas dependências subjacentes, consulte os tópicos a seguir no Guia de referência de [AWSSDKs e ferramentas](#):

- [AWS Política de manutenção de SDKs e ferramentas](#)

- [AWS Matriz de suporte de versões de SDKs e ferramentas](#)

Recursos adicionais

Além deste guia, os seguintes recursos online são importantes para desenvolvedores do AWS SDK for Java:

- [AWS SDK for Java Referência da API 2.x](#)
- [Blog de desenvolvedor Java](#)
- [Tópico de desenvolvimento Java em AWS re:Post](#)
- GitHub:
 - [Fonte de documentação do Developer Guide](#)
 - [Fonte do SDK](#)
- [AWS Biblioteca de exemplos de códigos do SDK](#)
- [@awsforjava \(Twitter\)](#)

Contribua com o SDK

Os desenvolvedores também podem contribuir com comentários por meio destes canais:

- Envie problemas em GitHub:
 - [Enviar problemas com a documentação do Developer Guide](#)
 - [Enviar problemas do SDK](#)
- [Participe de um bate-papo informal sobre o SDK no canal AWS SDK for Java 2.x gitter](#)

Comece com o AWS SDK for Java 2.x

O AWS SDK for Java 2.x fornece APIs Java para Amazon Web Services (AWS). Usando o SDK, você pode criar aplicativos Java que funcionam com Amazon S3, Amazon EC2, Amazon DynamoDB, e muito mais.

Este tutorial mostra como usar o [Apache Maven](#) para definir dependências para o SDK for Java 2.x e, em seguida, escrever um código que se conecta para fazer Amazon S3 o upload de um arquivo.

Siga estas etapas para concluir este tutorial:

- [Etapa 1: Configurar este tutorial](#)
- [Etapa 2: criar um projeto](#)
- [Etapa 3: escrever o código](#)
- [Etapa 4: Compilar e executar o aplicativo](#)

Etapa 1: Configurar este tutorial

Antes de iniciar este tutorial, você precisará fazer o seguinte:

- Permissão para acessar Amazon S3
- Um ambiente de desenvolvimento Java configurado para acessar Serviços da AWS usando o login único no AWS IAM Identity Center

Use as instruções em [Configuração básica](#) para se preparar para este tutorial. Depois de [configurar seu ambiente de desenvolvimento com acesso de login único](#) para o Java SDK e de ter uma [sessão ativa do portal de AWS acesso](#), continue com a Etapa 2 deste tutorial.

Etapa 2: criar um projeto

Para criar o projeto para este tutorial, você executa um comando Maven que solicita informações sobre como configurar o projeto. Depois que todas as entradas forem inseridas e confirmadas, o Maven conclui a construção do projeto criando pom.xml e criando arquivos Java stub.

1. Abra uma janela de terminal ou prompt de comando e navegue até um diretório de sua escolha, por exemplo, sua Home pasta Desktop ou.

2. Digite o seguinte comando no terminal e pressione `Enter`.

```
mvn archetype:generate \
-DarchetypeGroupId=software.amazon.awssdk \
-DarchetypeArtifactId=archetype-app-quickstart \
-DarchetypeVersion=2.20.43
```

3. Insira o valor listado na segunda coluna para cada solicitação.

Solicitação	Valor a ser inserido
Define value for property 'service':	s3
Define value for property 'httpClient' :	apache-client
Define value for property 'nativeImage' :	false
Define value for property 'credentialProvider'	identity-center
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. Depois que o último valor é inserido, o Maven lista as escolhas que você fez. Confirme inserindo `Y` ou insira novamente os valores `N` inserindo.

O Maven cria a pasta do projeto nomeada `getstarted` com base no `artifactId` valor que você inseriu. Dentro da `getstarted` pasta, encontre um `README.md` arquivo que você possa revisar, um `pom.xml` arquivo e um `src` diretório.

O Maven constrói a seguinte árvore de diretórios.

```
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
```

O exemplo a seguir mostra o conteúdo de um arquivo de `pom.xml` projeto.

pom.xml

A `dependencyManagement` seção `dependencies` tem uma dependência para o AWS SDK for Java 2.x dependências Amazon S3. O projeto usa o Java 1.8 devido ao `1.8` valor nas `maven.compiler.target` propriedades `maven.compiler.source` e.

```
<?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>2.20.43</aws.java.sdk.version> <----- SDK version
picked up from archetype 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> <----- S3 dependency
        <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>sso</artifactId> <----- Required for identity center authentication.
</dependency>

<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>ssooidc</artifactId> <----- Required for identity center authentication.
</dependency>

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

Etapa 3: escrever o código

O código a seguir mostra a App classe criada pelo Maven. O main método é o ponto de entrada no aplicativo, que cria uma instância da Handler classe e depois chama seu sendRequest método.

Classe App

```
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");
```

```
    }  
}
```

A `DependencyFactory` classe criada pelo Maven contém o método de `s3Client` fábrica que cria e retorna uma `S3Client` instância. A `S3Client` instância usa uma instância do cliente HTTP baseado em Apache. Isso ocorre porque você especificou `apache-client` quando o Maven solicitou qual cliente HTTP usar.

O `DependencyFactory` é mostrado no código a seguir.

Classe `DependencyFactory`

```
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();  
    }  
}
```

A `Handler` classe contém a lógica principal do seu programa. Quando uma instância de `Handler` é criada na `App` classe, o `DependencyFactory` fornece o cliente `S3Client` de serviço. Seu código usa a `S3Client` instância para chamar o serviço do Amazon S3.

O Maven gera a seguinte `Handler` classe com um `TODO` comentário. A próxima etapa do tutorial substitui o `TODO` por código.

Handler classe, gerada pelo Maven

```
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.
    }
}
```

Para preencher a lógica, substitua todo o conteúdo da Handler classe pelo código a seguir. O sendRequest método é preenchido e as importações necessárias são adicionadas.

Handler classe, implementada

O código primeiro cria um novo bucket do S3 com a última parte do nome gerado usando System.currentTimeMillis() para tornar o nome do bucket exclusivo.

Depois de criar o bucket no createBucket() método, o programa carrega um objeto usando o [putObject](#) método deS3Client. O conteúdo do objeto é uma string simples criada com o RequestBody.fromString método.

Finalmente, o programa exclui o objeto seguido pelo bucket no cleanUp método.

```
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");
}
}
```

Etapa 4: Compilar e executar o aplicativo

Depois que o projeto for criado e contiver a Handler classe completa, crie e execute o aplicativo.

1. Verifique se você tem uma sessão ativa do IAM Identity Center. Para fazer isso, execute o AWS Command Line Interface comando `aws sts get-caller-identity` e verifique a resposta. Se você não tiver uma sessão ativa, consulte [esta seção](#) para obter as instruções.
2. Abra uma janela de terminal ou de um prompt de comando e navegue até o diretório `getstarted` do projeto.
3. Use o comando a seguir para compilar seu projeto:

```
mvn clean package
```

4. Use o comando a seguir para executar o aplicativo.

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

Para visualizar o novo compartimento e o novo objeto que o programa cria, execute as etapas a seguir.

1. Em `Handler.java`, comente a linha `cleanUp(s3Client, bucket, key)` no `sendRequest` método e salve o arquivo.
2. Reconstrua o projeto executando `mvn clean package`
3. Execute novamente `mvn exec:java -Dexec.mainClass="org.example.App"` para carregar o objeto de texto mais uma vez.
4. Faça login no [console do S3](#) para ver o novo objeto no bucket recém-criado.

Depois de visualizar o arquivo, exclua o objeto e, em seguida, exclua o bucket.

Bem-sucedida

Se seu projeto Maven foi criado e executado sem erros, parabéns! Você compilou com sucesso seu primeiro aplicativo Java usando o SDK for Java 2.x.

Limpeza

Para limpar os recursos criados durante este tutorial, faça o seguinte:

- Se você ainda não tiver feito isso, no [console do S3](#), exclua todos os objetos e compartimentos criados quando você executou o aplicativo.
- No [IAMconsole](#), exclua o usuário do TestSDK.

Se você excluir esse usuário, remova também o conteúdo do `credentials` arquivo criado durante a configuração.

- Exclua a pasta do projeto (`getstarted`).

Próximas etapas

Agora que você entendeu o básico, saiba o seguinte:

- [Trabalhar com Amazon S3](#)
- [Trabalhando com outros Amazon Web Services](#), como [DynamoDB](#)[Amazon EC2](#), e [vários serviços de banco de dados](#)
- [Use o SDK](#)
- [Segurança para o AWS SDK for Java](#)

Configure o AWS SDK for Java 2.x

O AWS SDK for Java 2.x fornece APIs Java para Amazon Web Services (AWS). Usando o SDK, você pode criar aplicativos Java que funcionam com Amazon S3, Amazon EC2, Amazon DynamoDB, e muito mais.

Esta seção fornece informações sobre como configurar seu ambiente de desenvolvimento e projetos para usar AWS SDK for Java 2.x o.

Conteúdo deste capítulo

- [Configuração básica para trabalhar Serviços da AWS](#)
 - [Visão geral](#)
 - [Capacidade de login no portal de acesso AWS](#)
 - [Configurar o acesso de login único para o SDK](#)
 - [Faça login usando o AWS CLI](#)
 - [Instale o Java e uma ferramenta de construção](#)
 - [Opções adicionais de autenticação](#)
- [Configuração de ferramentas de construção](#)
 - [Configurar um projeto Apache Maven](#)
 - [Pré-requisitos](#)
 - [Criar um projeto Maven](#)
 - [Configurar o compilador Java para Maven](#)
 - [Declarar o SDK como dependência](#)
 - [Definir dependências para módulos do SDK](#)
 - [Desenvolver todo o SDK no projeto](#)
 - [Compilar o projeto](#)
 - [Configurar um projeto Gradle](#)
- [Configure um projeto GraalVM Native Image para o AWS SDK for Java](#)
 - [Pré-requisitos](#)
 - [Crie um projeto usando o arquétipo](#)
 - [Crie uma imagem nativa](#)

Configuração básica para trabalhar Serviços da AWS

Visão geral

Para desenvolver com sucesso aplicativos que acessem Serviços da AWS usando o AWS SDK for Java, as seguintes condições são necessárias:

- Você deve ser capaz de [entrar no portal de AWS acesso](#) disponível no AWS IAM Identity Center.
- As [permissões da função do IAM](#) configurada para o SDK devem permitir o acesso ao Serviços da AWS que seu aplicativo exige. As permissões associadas à política PowerUserAccessAWSgerenciada são suficientes para a maioria das necessidades de desenvolvimento.
- Um ambiente de desenvolvimento com os seguintes elementos:
 - [Arquivos de configuração compartilhados](#) que são configurados de pelo menos uma das seguintes formas:
 - O config arquivo contém as [configurações de login único do IAM Identity Center](#) para que o SDK possa obter credenciais. AWS
 - O credentials arquivo contém credenciais temporárias.
 - Uma [instalação do Java 8](#) ou posterior.
 - Uma [ferramenta de automação de construção](#), como [Maven](#) ou [Gradle](#).
 - Um editor de texto para trabalhar com código.
 - (Opcional, mas recomendado) Um IDE (ambiente de desenvolvimento integrado), como [IntelliJ IDEA](#), [Eclipse](#) ou [NetBeans](#)

Ao usar um IDE, você também pode integrar AWS Toolkit s para trabalhar com mais facilidade Serviços da AWS. Os [AWS Toolkit para IntelliJ](#) e [AWS Toolkit for Eclipse](#) são dois kits de ferramentas que você pode usar para desenvolvimento em Java.

- Uma sessão ativa do portal de AWS acesso quando você estiver pronto para executar seu aplicativo. Você usa o AWS Command Line Interface para [iniciar o processo de login no portal de acesso do IAM Identity Center](#). AWS

Important

As instruções nesta seção de configuração pressupõem que você ou a organização usam o IAM Identity Center. Se sua organização usa um provedor de identidade externo que funciona independentemente do IAM Identity Center, descubra como você pode obter credenciais temporárias para o SDK for Java usar. Siga [estas instruções](#) para adicionar credenciais temporárias ao `~/.aws/credentials` arquivo.

Se seu provedor de identidade adicionar credenciais temporárias automaticamente ao `~/.aws/credentials` arquivo, certifique-se de que o nome do perfil seja `[default]` para que você não precise fornecer um nome de perfil ao SDK ou AWS CLI.

Capacidade de login no portal de acesso AWS

O portal de AWS acesso é o local da web em que você faz login manualmente no IAM Identity Center. O formato do URL é `d-xxxxxxxxxx.awsapps.com/start` ou `your_subdomain.awsapps.com/start`. Se você não estiver familiarizado com o portal de AWS acesso, siga as orientações para acesso à conta no tópico de [autenticação do IAM Identity Center](#) no Guia de referência de AWS SDKs e ferramentas.

Configurar o acesso de login único para o SDK

Depois de concluir a Etapa 2 na [seção de acesso programático](#) para que o SDK use a autenticação do IAM Identity Center, seu sistema deve conter os seguintes elementos.

- O AWS CLI, que você usa para iniciar uma [sessão do portal de AWS acesso](#) antes de executar seu aplicativo.
- Um `~/.aws/config` arquivo que contém um [perfil padrão](#). O SDK for Java usa a configuração do provedor de token SSO do perfil para adquirir credenciais antes de enviar solicitações para AWS. O valor `sso_role_name`, que é um perfil do IAM conectado a um conjunto de permissões do Centro de Identidade do IAM, deve permitir o acesso a Serviços da AWS usado na aplicação.

O `config` arquivo de exemplo a seguir mostra um perfil padrão configurado com a configuração do provedor de token SSO. A configuração `sso_session` do perfil se refere à seção chamada `sso-session`. A seção `sso-session` contém configurações para iniciar uma sessão do portal de acesso da AWS.

```
[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
```

Para obter mais detalhes sobre as configurações usadas na configuração do provedor de token SSO, consulte Configuração do provedor de [token SSO no Guia](#) de referência de AWS SDKs e ferramentas.

Se seu ambiente de desenvolvimento não estiver configurado para acesso programático conforme mostrado anteriormente, siga a [Etapa 2 no Guia de referência dos SDKs](#).

Faça login usando o AWS CLI

Antes de executar um aplicativo que acessa Serviços da AWS, você precisa de uma sessão ativa do portal de AWS acesso para que o SDK use a autenticação do IAM Identity Center para resolver as credenciais. Execute o seguinte comando no AWS CLI para entrar no portal de AWS acesso.

```
aws sso login
```

Como você tem uma configuração de perfil padrão, não precisa chamar o comando com uma opção `--profile`. Se a configuração do provedor de token de SSO estiver usando um perfil nomeado, o comando será `aws sso login --profile named-profile`.

Para testar se você já tem uma sessão ativa, execute o comando da AWS CLI a seguir.

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

A resposta a esse comando deve relatar a conta do Centro de Identidade do IAM e o conjunto de permissões configurados no arquivo compartilhado `config`.

Note

Se você já tiver uma sessão ativa do portal de acesso da AWS e executar `aws sso login`, não será necessário fornecer credenciais.

No entanto, você verá uma caixa de diálogo solicitando permissão botocore para acessar suas informações. botocoreé a base para AWS CLI o.

Selecione Permitir para autorizar o acesso às suas informações para o AWS CLI SDK for Java.

Instale o Java e uma ferramenta de construção

Seu ambiente de desenvolvimento precisa do seguinte:

- Java 8 ou posterior. AWS SDK for Java[Funciona com o Oracle Java SE Development Kit e com distribuições do Open Java Development Kit \(OpenJDK\) Amazon Corretto, como Red Hat OpenJDK e JDK. AdoptOpen](#)
- Uma ferramenta de compilação ou IDE compatível com o Maven Central, como Apache Maven, Gradle ou IntelliJ.
 - Para obter informações sobre como instalar e usar o Maven, consulte <http://maven.apache.org/>.
 - Para obter informações sobre como instalar e usar o Gradle, consulte <https://gradle.org/>.
 - [Para obter informações sobre como instalar e usar o IntelliJ IDEA, consulte https://www.jetbrains.com/idea/](https://www.jetbrains.com/idea/).

Opções adicionais de autenticação

Para obter mais opções de autenticação para o SDK, como o uso de perfis e variáveis de ambiente, consulte o capítulo de [configuração](#) no Guia de referência de AWS SDKs e ferramentas.

Configuração de ferramentas de construção

Configurar um projeto Apache Maven

Você pode usar o [Apache Maven](#) para configurar e criar AWS SDK for Java projetos ou para criar o próprio SDK.

Pré-requisitos

Para usar o AWS SDK for Java com o Maven, você precisa do seguinte:

- Java 8.0 ou posterior. Você pode fazer download do software do Java SE Development Kit mais recente em <http://www.oracle.com/technetwork/java/javase/downloads/>. O AWS SDK for Java também funciona com o [OpenJDK](#) e o Amazon Corretto, uma distribuição do Open Java Development Kit (OpenJDK). Faça download da versão mais recente do OpenJDK em <https://openjdk.java.net/install/index.html>. Baixe a versão Amazon Corretto 8 ou Amazon Corretto 11 mais recente [da Corretto página](#).
- Apache Maven. Se você precisar instalar o Maven, acesse <http://maven.apache.org/> para baixá-lo.

Criar um projeto Maven

Para criar um projeto Maven a partir da linha de comando, execute o comando a seguir em um terminal ou janela do prompt de comando.

```
mvn -B archetype:generate \
-DarchetypeGroupId=software.amazon.awssdk \
-DarchetypeArtifactId=archetype-lambda -Dservice=s3 -Dregion=US_WEST_2 \
-DarchetypeVersion=2.X.X \
-DgroupId=com.example.myapp \
-DartifactId=myapp
```

Note

Substitua com.example.myapp pelo namespace do pacote completo do aplicativo. Substitua myapp pelo nome do projeto. Esse será o nome do diretório do projeto.

Para usar a versão mais recente do arquétipo, substitua **2.X.X** pela versão [mais recente](#) do Maven central.

Esse comando cria um projeto Maven usando o kit de ferramentas de modelagem de arquétipos. O arquétipo gera o andaime para um AWS Lambda projeto de manipulador de funções. Esse arquétipo de projeto é pré-configurado para compilar com o Java SE 8 e inclui uma dependência da versão do SDK for Java 2.x especificada com. -DarchetypeVersion

Para obter mais informações sobre como criar e configurar projetos do Maven, consulte o [Maven Getting Started Guide](#).

Configurar o compilador Java para Maven

Se você criou seu projeto usando o arquétipo do AWS Lambda projeto conforme descrito anteriormente, a configuração do compilador Java já foi feita para você.

Para verificar se a configuração está presente, primeiro abra o arquivo pom.xml da pasta do projeto que você criou (por exemplo, myapp) ao executar o comando anterior. Veja as linhas 11 e 12 para confirmar a configuração da versão do compilador Java desse projeto Maven, e a inclusão necessária do plugin do compilador Maven nas linhas 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>
```

Se você criar seu projeto com um arquétipo diferente ou por outro método, certifique-se de que o plug-in do compilador Maven faz parte da compilação e que suas propriedades de origem e destino estão definidas como 1.8 no pom.xml arquivo.

Consulte o snippet anterior para ver uma maneira de definir essas configurações necessárias.

Como opção, você pode configurar o compilador em linha com a declaração do plugin, como a seguir.

```
<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>
```

Declarar o SDK como dependência

Para usar o AWS SDK for Java no seu projeto, você precisa declará-lo como uma dependência no arquivo pom.xml do projeto.

Se você criou seu projeto usando o arquétipo do projeto, conforme descrito anteriormente, a versão mais recente do SDK já está configurada como uma dependência em seu projeto.

O arquétipo gera uma dependência do artefato BOM (lista de materiais) para o ID do grupo software.amazon.awssdk Com um BOM, você não precisa especificar a versão do maven para dependências de artefatos individuais que compartilham o mesmo ID de grupo.

Se você criou seu projeto do Maven de maneira diferente, configure a versão mais recente do SDK para seu projeto garantindo que o arquivo pom.xml contenha o seguinte.

```
<project>
<properties>
    <aws.java.sdk.version>2.X.X</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

Substitua **2.X.X** no pom.xml arquivo pela [versão mais recente do AWS SDK for Java 2.x](#)

Definir dependências para módulos do SDK

Agora que você configurou o SDK, pode adicionar dependências para um ou mais módulos do AWS SDK for Java para usar no projeto.

Embora você possa especificar o número da versão de cada componente, não é necessário porque você já declarou a versão do SDK na dependencyManagement seção usando o artefato da lista de materiais. Para carregar uma versão diferente de um determinado módulo, especifique um número de versão para sua dependência.

Se você criou seu projeto usando o arquétipo do projeto conforme descrito anteriormente, seu projeto já está configurado com várias dependências. Isso inclui dependências para manipuladores de AWS Lambda funções e Amazon S3, conforme a seguir.

```
<project>
  <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>url-connection-client</artifactId>
    </dependency>
```

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-lambda-java-core</artifactId>
    <version>${aws.lambda.java.version}</version>
</dependency>
</dependencies>
</project>
```

Note

No pom.xml exemplo acima, as dependências são de diferentes groupId's. A s3 dependência é desoftware.amazon.awssdk, enquanto a aws-lambda-java-core dependência é de com.amazonaws. A configuração do gerenciamento de dependências do BOM afeta os artefatos para software.amazon.awssdk, portanto, é necessária uma versão para o aws-lambda-java-core artefato.

Para o desenvolvimento de manipuladores de funções Lambda usando o SDK for Java 2.xaws-lambda-java-core, é a dependência correta. No entanto, se seu aplicativo precisar gerenciar recursos do Lambda, usando operações com `listFunctions`, `deleteFunction`, `invokeFunction` e `createFunction`, seu aplicativo exigirá a seguinte dependência.

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

Note

A s3 dependência exclui as dependências the netty-nio-client e apache-client transitive. No lugar de qualquer um desses clientes HTTP, o arquétipo inclui a url-connection-client dependência, que ajuda a [reduzir a latência de inicialização das funções](#). AWS Lambda

Adicione os módulos ao seu projeto para obter os recursos AWS service (Serviço da AWS) e os recursos necessários para seu projeto. Os módulos (dependências) que são gerenciados pelo AWS SDK for Java BOM estão listados no repositório [central do Maven](#).

Note

Para determinar quais dependências você precisa para o projeto, veja o arquivo pom.xml de um exemplo de código. [Por exemplo, se você estiver interessado nas dependências do serviço DynamoDB, veja esse exemplo no Repositório de exemplos de AWS código em GitHub](#) (Procure o pom.xml arquivo em [/javav2/example_code/dynamodb](#).)

Desenvolver todo o SDK no projeto

Para otimizar o aplicativo, recomendamos que você extraia apenas os componentes de que precisa, em vez de todo o SDK. No entanto, para criar o AWS SDK for Java inteiro no projeto, declare-o no arquivo pom.xml, da seguinte forma.

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

Compilar o projeto

Depois de configurar o arquivo pom.xml, você poderá usar o Maven para desenvolver o projeto.

Para criar o projeto Maven pela linha de comando, abra uma janela de terminal ou prompt de comando, navegue até o diretório dele (por exemplo, myapp), insira ou cole o comando abaixo e pressione Enter ou Return.

```
mvn package
```

Isso criará um único arquivo (JAR) .jar no diretório target (por exemplo, myapp/target). Esse JAR conterá todos os módulos SDK especificados como dependências no arquivo pom.xml.

Configurar um projeto Gradle

Você pode usar o [Gradle](#) para configurar e criar AWS SDK for Java projetos.

As etapas iniciais no exemplo a seguir vêm do [guias de introdução do Gradle](#) para a versão 8.4. Se você usar uma versão diferente, seus resultados poderão ser um pouco diferentes.

Para criar um aplicativo Java com o Gradle (linha de comando)

1. Crie um diretório para armazenar seu projeto. Neste exemplo, demo é o nome do diretório.
2. Dentro do demo diretório, execute o gradle init comando e forneça os valores destacados em vermelho, conforme mostrado na saída da linha de comando a seguir. Para a apresentação, escolhemos o Kotlin como a linguagem DSL do script de construção, mas um exemplo completo do Groovy também é mostrado no final deste tópico.

```
> gradle init
Starting a Gradle Daemon (subsequent builds will be faster)

Select type of project to generate:
1: basic
2: application
3: library
4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
1: C++
2: Groovy
3: Java
4: Kotlin
5: Scala
6: Swift
Enter selection (default: Java) [1..6] 3

Generate multiple subprojects for application? (default: no) [yes, no] no
Select build script DSL:
1: Kotlin
2: Groovy
Enter selection (default: Kotlin) [1..2] <Enter>

Select test framework:
1: JUnit 4
2: TestNG
3: Spock
4: JUnit Jupiter
Enter selection (default: JUnit Jupiter) [1..4] 4
```

```
Project name (default: demo): <Enter>
Source package (default: demo): <Enter>
Enter target version of Java (min. 7) (default: 11): <Enter>
Generate build using new APIs and behavior (some features may change in the next
minor release)? (default: no) [yes, no] <Enter>

> Task :init
To learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.4/
samples/sample_building_java_applications.html

BUILD SUCCESSFUL in 3m 43s
2 actionable tasks: 2 executed
```

3. Depois que a `init` tarefa for concluída, o demo diretório conterá a seguinte estrutura em árvore. Examinaremos mais de perto o arquivo de compilação principal `build.gradle.kts` (destacado em vermelho) na próxima seção.

```
### app
#   ### build.gradle.kts
#   ### src
#       ### main
#           #   ### java
#           #   #   ### demo
#           #   #       ### App.java
#           #   ### resources
#           ### test
#               ### java
#               #   ### demo
#               #       ### AppTest.java
#               ### resources
### gradle
#   ### wrapper
#       ### gradle-wrapper.jar
#       ### gradle-wrapper.properties
### gradlew
### gradlew.bat
### settings.gradle.kts
```

O `build.gradle.kts` arquivo contém o seguinte conteúdo em andamento.

```
/*
```

```
* This file was generated by the Gradle 'init' task.  
*  
* This generated file contains a sample Java application project to get you  
started.  
* For more details on building Java & JVM projects, please refer to https://docs.gradle.org/8.4/userguide/building\_java\_projects.html in the Gradle  
documentation.  
*/  
  
plugins {  
    // Apply the application plugin to add support for building a CLI application  
    // in Java.  
    application  
}  
  
repositories {  
    // Use Maven Central for resolving dependencies.  
    mavenCentral()  
}  
  
dependencies {  
    // Use JUnit Jupiter for testing.  
    testImplementation("org.junit.jupiter:junit-jupiter:5.9.3")  
  
    testRuntimeOnly("org.junit.platform:junit-platform-launcher")  
  
    // This dependency is used by the application.  
    implementation("com.google.guava:guava:32.1.1-jre")  
}  
  
// Apply a specific Java toolchain to ease working on different environments.  
java {  
    toolchain {  
        languageVersion.set(JavaLanguageVersion.of(11))  
    }  
}  
  
application {  
    // Define the main class for the application.  
    mainClass.set("demo.App")  
}  
  
tasks.named<Test>("test") {  
    // Use JUnit Platform for unit tests.
```

```
    useJUnitPlatform()  
}
```

4. Use o arquivo de compilação do Gradle em andaime como base para seu projeto. AWS

- Para gerenciar as dependências do SDK para seu projeto Gradle, adicione a lista de materiais (BOM) do Maven AWS SDK for Java 2.x à dependencies seção do arquivo. build.gradle.kts

```
...  
dependencies {  
    implementation(platform("software.amazon.awssdk:bom:2.21.1"))  
    // With the bom declared, you specify individual SDK dependencies without a  
    // version.  
    ...  
}  
...
```

 Note

Neste exemplo de arquivo de compilação, substitua 2.21.1 pela versão mais recente do SDK for Java 2.x. Encontre a versão mais recente disponível no [repositório central do Maven](#).

- Especifique os módulos do SDK que seu aplicativo precisa na dependencies seção. Como exemplo, o seguinte adiciona uma dependência do Amazon Simple Storage Service.

```
...  
dependencies {  
    implementation(platform("software.amazon.awssdk:bom:2.21.1"))  
    implementation("software.amazon.awssdk:s3")  
    ...  
}  
...
```

O Gradle resolve automaticamente a versão correta das dependências declaradas usando as informações do BOM.

Os exemplos a seguir mostram arquivos de compilação completos do Gradle nas DSLs Kotlin e Groovy. O arquivo de compilação contém dependências para Amazon S3, autenticação, registro e teste. A versão de origem e destino do Java é a versão 11.

Kotlin DSL (build.gradle.kts)

```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java application project to get you
 * started.
 * For more details on building Java & JVM projects, please refer to https://
docs.gradle.org/8.4/userguide/building_java_projects.html in the Gradle
documentation.
 */

plugins {
    // Apply the application plugin to add support for building a CLI application in
    Java.
    application
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.20.56"))
    implementation("software.amazon.awssdk:s3")
    implementation("software.amazon.awssdk:sso")
    implementation("software.amazon.awssdk:ssoidc")
    implementation(platform("org.apache.logging.log4j:log4j-bom:2.20.0"))
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")
    implementation("org.apache.logging.log4j:log4j-1.2-api")
    testImplementation(platform("org.junit:junit-bom:5.10.0"))
    testImplementation("org.junit.jupiter:junit-jupiter")
}

// Apply a specific Java toolchain to ease working on different environments.
java {
    toolchain {
        languageVersion.set(JavaLanguageVersion.of(11))
```

```
    }

}

application {
    // Define the main class for the application.
    mainClass.set("demo.App")
}

tasks.named<Test>("test") {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
}
```

Groovy DSL (build.gradle)

```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java application project to get you
 * started.
 * For more details on building Java & JVM projects, please refer to https://
docs.gradle.org/8.4/userguide/building_java_projects.html in the Gradle
documentation.
 */

plugins {
    // Apply the application plugin to add support for building a CLI application in
Java.
    id 'application'
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {
    implementation platform('software.amazon.awssdk:bom:2.21.1')
    implementation 'software.amazon.awssdk:s3'
    implementation 'software.amazon.awssdk:sso'
    implementation 'software.amazon.awssdk:ssoidc'
    implementation platform('org.apache.logging.log4j:log4j-bom:2.20.0')
    implementation 'org.apache.logging.log4j:log4j-slf4j2-impl'
```

```
implementation 'org.apache.logging.log4j:log4j-1.2-api'
testImplementation platform('org.junit:junit-bom:5.10.0')
testImplementation 'org.junit.jupiter:junit-jupiter'
}

// Apply a specific Java toolchain to ease working on different environments.
java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(11)
    }
}

application {
    // Define the main class for the application.
    mainClass = 'demo_groovy.App'
}

tasks.named('test') {
    // Use JUnit Platform for unit tests.
    useJUnitPlatform()
}
```

Para ver as próximas etapas, consulte o guia de primeiros passos no site do Gradle para obter instruções sobre como [criar e executar um aplicativo Gradle](#).

Configure um projeto GraalVM Native Image para oAWS SDK for Java

Com as versões 2.16.1 e posteriores, oAWS SDK for Java fornece out-of-the-box suporte para aplicativos GraalVM Native Image. Use o arquétipo do archetype-app-quickstart Maven para configurar um projeto com suporte de imagem nativo integrado.

Pré-requisitos

- Conclua as etapas em [Configurando oAWS SDK for Java 2.x](#).
- Instale o [GraalVM Native Image](#).

Crie um projeto usando o arquétipo

Para criar um projeto Maven com suporte de imagem nativo integrado, em uma janela de terminal ou prompt de comando, use o comando a seguir.

 Note

com.example.mynativeimageappSubstitua pelo namespace completo do pacote do seu aplicativo. Também mynnativemageapp substitua pelo nome do seu projeto. Esse será o nome do diretório do projeto.

```
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
```

Esse comando cria um projeto Maven configurado com dependências para o AWS SDK for Java Amazon S3, e o clienteApacheHttpClient HTTP. Ele também inclui uma dependência para o [plugin GraalVM Native Image Maven](#), para que você possa criar imagens nativas usando o Maven.

Para incluir dependências para outroAmazon Web Services, defina o valor do -Dservice parâmetro como o ID do artefato desse serviço. Os exemplos incluem dynamodb, comprehend e pinpoint. Para obter uma lista completa de IDs de artefatos, consulte a lista de dependências gerenciadas para [software.amazon.awssdk no Maven Central](#).

Para usar um cliente HTTP assíncrono, defina o -DhttpClient parâmetro comonetty-nio-client. Para usarUrlConnectionHttpClient como cliente HTTP síncrono em vez de apache-client, defina o -DhttpClient parâmetro comourl-connection-client.

Crie uma imagem nativa

Depois de criar o projeto, execute o seguinte comando no diretório do seu projeto, por exemplo mynativeimageapp:

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

Isso cria um aplicativo de imagem nativo no diretório, por exemplo, target/mynativeimageapp.

Usar a AWS SDK for Java 2.x

Depois de concluir as etapas em [Configurando o SDK](#), você está pronto para fazer solicitações para AWS serviços como Amazon S3, DynamoDB, IAM, Amazon EC2 e muito mais.

Trabalhe com clientes de serviços

Crie um cliente de serviço

Para fazer uma solicitação a um AWS service (Serviço da AWS), você deve primeiro instanciar um cliente de serviço para esse serviço usando o método estático de fábrica, `builder()`. O `builder()` método retorna um `builder` objeto que permite personalizar o cliente do serviço. Os métodos setter fluentes retornam o objeto `builder`, de maneira que você possa vincular as chamadas ao método por conveniência e obter um código mais legível. Depois de configurar as propriedades desejadas, chame `build()` método para criar o cliente.

Como exemplo, o trecho de código a seguir instancia um `Ec2Client` objeto como cliente de serviço para o Amazon EC2.

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

Note

Os clientes de serviço no SDK devem ser livres de thread. Para obter uma melhor performance, trate-os como objetos de longa duração. Cada cliente tem o próprio recurso do grupo de conexões, que é liberado quando o lixo do cliente é coletado.

Um objeto cliente de serviço é imutável, então você deve criar um novo cliente para cada serviço para o qual você faz solicitações, ou se quiser usar uma configuração diferente para fazer solicitações para o mesmo serviço.

Especificando o `Region` no serviço, o construtor de clientes não é necessário para todos os AWS serviços; no entanto, é uma prática recomendada definir a região para as chamadas de API que você faz em seus aplicativos. Veja [AWS seleção de região](#) para obter mais informações.

Configuração padrão do cliente

Os compiladores de cliente têm outro método de fábrica chamado `create()`. Esse método cria um cliente de serviço com a configuração padrão. Ele usa a cadeia de provedores padrão para carregar as credenciais e o Região da AWS. Se as credenciais ou a região não puderem ser determinadas a partir do ambiente em que o aplicativo está sendo executado, a chamada para `create()` falha. Veja [Usando credenciais](#) e [Seleção de região](#) para obter mais informações sobre como o SDK determina as credenciais e a região a serem usadas.

Por exemplo, o trecho de código a seguir instancia um `DynamoDbClient` objeto como cliente de serviço para o Amazon DynamoDB:

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

Configurar clientes de serviço

Para personalizar a configuração de um cliente de serviço, use os setters no `builder()` método de fábrica. Para maior comodidade e para criar um código mais legível, encadeie os métodos para definir várias opções de configuração.

O exemplo a seguir mostra um `S3Client` que é configurado com várias configurações personalizadas.

```
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();
```

Faça solicitações

Use o cliente de serviço para fazer solicitações ao correspondente AWS service (Serviço da AWS).

Por exemplo, esse trecho de código mostra como criar um `RunInstancesRequest` objeto para criar uma nova instância do Amazon EC2:

```
// Create the request by using the fluid setter methods of the request builder.  
RunInstancesRequest runInstancesRequest = RunInstancesRequest.builder()  
    .imageId(amiId)  
    .instanceType(InstanceType.T1_MICRO)  
    .maxCount(1)  
    .minCount(1)  
    .build();  
  
// Use the configured request with the service client.  
ec2Client.runInstances(runInstancesRequest);
```

Lidar com respostas

Você usa um manipulador de respostas para processar a resposta de volta do AWS service (Serviço da AWS).

Por exemplo, esse trecho de código mostra como criar um `RunInstancesResponse` objeto para lidar com a resposta do Amazon EC2 imprimindo o `instanceId` para a nova instância da solicitação acima:

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

Feche o cliente de serviço

Como prática recomendada, você deve usar um cliente de serviço para várias chamadas de serviço de API durante a vida útil de um aplicativo. No entanto, se você precisar de um cliente de serviço para uso único ou não precisar mais do cliente de serviço, feche-o.

Ligue para o `close()` método quando o cliente do serviço não é mais necessário para liberar recursos.

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

Se você precisar de um cliente de serviço para uso único, poderá instanciar o cliente de serviço como um recurso em um `try`-`catch` declaração - com recursos. Os clientes de serviços implementam a interface [AutoCloseable](#), para que o JDK chame automaticamente o `close()` método no final da declaração.

O exemplo a seguir demonstra como usar um cliente de serviço para uma chamada única. O `StsClient` que chama o AWS Security Token Service é fechado após retornar o ID da conta.

```
import software.amazon.awssdk.services.sts.StsClient;

String getAccountID() {
    try (StsClient stsClient = StsClient.create()) {
        return stsClient.getCallerIdentity().account();
    }
}
```

Lidar com exceções

O SDK usa exceções de tempo de execução (ou não verificadas), fornecendo controle refinado sobre o tratamento de erros e garantindo que o tratamento de exceções seja escalonado com seu aplicativo.

Um [SdkServiceException](#), ou uma de suas subclasses, é a forma mais comum de exceção que o SDK lançará. Essas exceções representam respostas do AWS serviço. Você também pode lidar com um [SdkClientException](#), que ocorre quando há um problema no lado do cliente (ou seja, em seu ambiente de desenvolvimento ou aplicativo), como uma falha na conexão de rede.

Esse trecho de código demonstra uma maneira de lidar com exceções de serviço ao fazer upload de um arquivo para Amazon S3. O código de exemplo captura as exceções do cliente e do servidor, registra os detalhes e existe o aplicativo.

```
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 se) {
    System.err.println("Service exception thrown.");
    System.err.println(se.awsErrorDetails().errorMessage());
} catch (SdkClientException ce){
    System.err.println("Client exception thrown.");
    System.err.println(ce.getMessage());
} finally {
    System.exit(1);
}
```

Veja [Lidando com exceções](#) para obter mais informações.

Use garçons

Algumas solicitações demoram para serem processadas, como a criação de uma nova tabela no DynamoDB ou criando um novo Amazon S3 balde. Para garantir que o recurso esteja pronto antes que seu código continue sendo executado, use um Garçom.

Por exemplo, esse trecho de código cria uma nova tabela (“MyTable”) no DynamoDB, espera que a mesa esteja em um ACTIVE status e, em seguida, imprime a resposta:

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

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

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

Veja [Usando garçons](#) para obter mais informações.

Clientes HTTP

Você pode alterar a configuração padrão para clientes HTTP em aplicativos que você cria com o AWS SDK for Java. Para obter informações sobre como definir clientes e configurações HTTP, consulte[Configuração HTTP](#).

Tentativas

Você pode alterar as configurações padrão para novas tentativas em seus clientes de serviço, incluindo o modo de repetição e a estratégia de recuo. Para obter mais informações, consulte[aRetryPolicy classe](#)no AWS SDK for JavaReferência da API.

Para obter mais informações sobre novas tentativas em AWSserviços, consulte[Tentativas de erro e recuo exponencial em AWS](#).

Tempo limite atingido

Você pode configurar tempos limite para cada um dos seus clientes de serviço usando o `apiCallTimeout` e `apiCallAttemptTimeout`setters. `OapiCallTimeout`configuração é a quantidade de tempo para permitir que o cliente conclua a execução de uma chamada de API. `OapiCallAttemptTimeout`configuração é a quantidade de tempo de espera até que a solicitação HTTP seja concluída antes de desistir.

Para obter mais informações, consulte[apiCallTimeout](#) e `apiCallAttemptTimeout`no AWS SDK for JavaReferência da API.

Interceptores de execução

Você pode escrever um código que intercepte a execução de suas solicitações e respostas de API em diferentes partes do ciclo de vida da solicitação/resposta. Isso permite que você publique métricas, modifique uma solicitação em andamento, depure o processamento da solicitação, visualize exceções e muito mais. Para obter mais informações, consulte[aExecutionInterceptor interface](#)no AWS SDK for JavaReferência da API.

Informações adicionais

- Para obter exemplos completos dos trechos de código acima, consulte[Trabalhando comAmazon DynamoDB](#),[Trabalhando comAmazon EC2](#), e[Trabalhando comAmazon S3](#).

Forneça credenciais temporárias ao SDK

Antes de fazer uma solicitação para Amazon Web Services usando o AWS SDK for Java 2.x, o SDK assina criptograficamente as credenciais temporárias emitidas pela AWS. Para acessar as credenciais temporárias, o SDK recupera os valores de configuração verificando vários locais.

Este tópico discute várias maneiras de habilitar o SDK para acessar credenciais temporárias.

Tópicos

- [Usar credenciais temporárias](#)
- [Cadeia de provedores de credenciais padrão](#)
- [Use um provedor de credenciais específico ou uma cadeia de fornecedores](#)
- [Use perfis](#)
- [Carregar credenciais temporárias de um processo externo](#)
- [Forneça credenciais temporárias em código](#)
- [Configurar IAM funções para Amazon EC2](#)

Usar credenciais temporárias

Para maior segurança, AWS recomenda que você configure o SDK para Java para [usar credenciais temporárias](#) em vez de credenciais duradouras. As credenciais temporárias consistem em chaves de acesso (ID da chave de acesso e chave de acesso secreta) e um token de sessão. Recomendamos que você [configurar o SDK](#) para obter automaticamente credenciais temporárias, já que o processo de atualização do token é automático. Você pode, no entanto, [forneça credenciais temporárias ao SDK](#) diretamente.

Configuração do IAM Identity Center

Quando você configura o SDK para usar o acesso de login único do IAM Identity Center, conforme descrito em [the section called “Configuração básica”](#) neste guia, o SDK usa automaticamente credenciais temporárias.

O SDK usa o token de acesso do IAM Identity Center para obter acesso à função do IAM que está configurada com `osso_role_name` configurando em seu `config` arquivo. O SDK assume essa função do IAM e recupera credenciais temporárias para serem usadas em AWS service (Serviço da AWS) solicitações.

Para obter mais detalhes sobre como o SDK obtém credenciais temporárias da configuração, consulte[Entendendo a autenticação do IAM Identity Center](#)seção do AWS Guia de referência de SDKs e ferramentas.

Recuperar de AWS portal de acesso

Como alternativa à configuração de login único do IAM Identity Center, você pode copiar e usar as credenciais temporárias disponíveis no AWS portal de acesso. É possível usar as credenciais temporárias em um perfil ou usá-las como valores para propriedades do sistema e variáveis de ambiente.

Configurar um arquivo de credenciais local para credenciais temporárias

1. [Crie um arquivo de credenciais compartilhado](#)
2. No arquivo de credenciais, cole o seguinte texto de espaço reservado até colar as credenciais temporárias de trabalho.

```
[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. Salve o arquivo. O arquivo `~/.aws/credentials` agora deve existir em seu sistema de desenvolvimento local. Esse arquivo contém [o perfil \[padrão\]](#) que o SDK para Java usa se um perfil nomeado específico não for especificado.
4. [Faça login no AWS portal de acesso](#)
5. Siga estas instruções sob o[Atualização manual de credenciais](#) cabeçalho para copiar as credenciais da função do IAM do AWS portal de acesso.
 - a. Para a etapa 4 nas instruções vinculadas, escolha o nome da função do IAM que concede acesso às suas necessidades de desenvolvimento. Essa função normalmente tem um nome como `PowerUserAccess` ou `Desenvolvedor`.
 - b. Para a etapa 7, selecione a Adicione manualmente um perfil ao seu AWS arquivo de credenciais opção e copie o conteúdo.
6. Cole as credenciais copiadas em seu local `credentials` arquivo e remova o nome do perfil gerado. Seu arquivo deve ser semelhante ao seguinte.

```
[default]
```

```
aws_access_key_id=AKIAIOSFODNN7EXAMPLE  
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY  
aws_session_token=IQoJb3JpZ2luX2I0oJb3JpZ2luX2I0oJb3JpZ2luX2I0oJb3JpZ2luX2I0oJb3JpZVERYLONGTOKEN
```

7. Salve o arquivo credentials.

Quando o SDK para Java cria um cliente de serviço, ele acessa essas credenciais temporárias e as usa em cada solicitação. As configurações do perfil do IAM escolhidas na etapa 5a determinam [por quanto tempo as credenciais temporárias são válidas](#). A duração máxima é de doze horas.

Depois que as credenciais temporárias expirarem, repita as etapas de 4 a 7.

Cadeia de provedores de credenciais padrão

A cadeia de fornecedores de credenciais padrão é implementada pelo [DefaultCredentialsProvider](#) classe. Ele verifica sequencialmente cada local onde você pode definir a configuração padrão para fornecer credenciais temporárias e, em seguida, seleciona a primeira que você definir.

Para usar a cadeia de provedores de credenciais padrão para fornecer credenciais temporárias que são usadas em seu aplicativo, crie um criador de clientes de serviço, mas não especifique um provedor de credenciais. O trecho de código a seguir cria um `DynamoDbEnhancedClient` que usa a cadeia de fornecedores de credenciais padrão para localizar e recuperar as configurações padrão.

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

Ordem de recuperação das configurações de credenciais

A cadeia de fornecedores de credenciais padrão do SDK para Java 2.x pesquisa a configuração em seu ambiente usando uma sequência predefinida.

1. Propriedades do sistema Java

- O SDK usa a classe `SystemPropertyCredentialsProvider` para carregar credenciais temporárias do `aws.accessKeyId`, `aws.secretAccessKey`, e `aws.sessionToken` das propriedades do sistema Java.

Note

Para obter informações sobre como definir as propriedades do sistema Java, consulte [Propriedades do sistema](#) tutorial sobre o oficial [Tutoriais de Javasite](#).

2. Variáveis de ambiente

- O SDK usa a classe [EnvironmentVariableCredentialsProvider](#) para carregar credenciais temporárias do AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, e AWS_SESSION_TOKEN variáveis de ambiente.

3. Token de identidade da Web de AWS Security Token Service

- O SDK usa a classe [WebIdentityTokenFileCredentialsProvider](#) para carregar credenciais temporárias das propriedades do sistema Java ou das variáveis de ambiente.

4. O compartilhado `credentials` se configura pastas

- O SDK usa a classe [ProfileCredentialsProvider](#) para carregar as configurações de login único do IAM Identity Center ou credenciais temporárias do [default] perfil no compartilhado `credentials` se configura arquivos.

O AWS guia de referência de SDKs e ferramentas tem [informações detalhadas](#) sobre como o SDK para Java funciona com o token de login único do IAM Identity Center para obter credenciais temporárias que o SDK usa para chamar serviços da AWS.

Note

O `credentials` se configura arquivos são compartilhados por vários AWS SDKs e ferramentas. Para obter mais informações, consulte [Os arquivos .aws/credentials e .aws/config](#) no AWS Guia de referência de SDKs e ferramentas.

5. Amazon ECS credenciais de contêiner

- O SDK usa a classe [ContainerCredentialsProvider](#) para carregar credenciais temporárias do AWS_CONTAINER_CREDENTIALS_RELATIVE_URI variável de ambiente do sistema.

6. Amazon EC2 credenciais fornecidas pela função IAM da instância

- O SDK usa a classe [InstanceProfileCredentialsProvider](#) para carregar credenciais temporárias do Amazon EC2 serviço de metadados.

Use um provedor de credenciais específico ou uma cadeia de fornecedores

Como alternativa à cadeia de provedores de credenciais padrão, você pode especificar qual provedor de credenciais o SDK deve usar. Quando você fornece um provedor de credenciais específico, o SDK ignora o processo de verificação de vários locais, o que reduz um pouco o tempo de criação de um cliente de serviço.

Por exemplo, se você definir sua configuração padrão usando variáveis de ambiente, forneça um [EnvironmentVariableCredentialsProvider](#) objeto ao `credentialsProvider` método no construtor do cliente de serviço, como no trecho de código a seguir.

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

Para obter uma lista completa de provedores de credenciais e cadeias de fornecedores, consulte Todas as classes de implementação conhecidas em [AwsCredentialsProvider](#).

 Note

Você pode usar seu próprio provedor de credenciais ou cadeias de provedores implementando o `AwsCredentialsProvider` interface.

Use perfis

Usando o compartilhado `config.credentials` arquivo, você pode configurar vários perfis. Isso permite que seu aplicativo use vários conjuntos de configuração de credenciais. O [default] perfil foi mencionado anteriormente. O SDK usa o [ProfileCredentialsProvider](#) classe para carregar configurações de perfis definidos no compartilhado `credentials` arquivo.

O trecho de código a seguir demonstra como criar um cliente de serviço que usa as configurações definidas como parte do perfil chamado `my_profile`.

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

```
.build();
```

Definir um perfil diferente como padrão

Para definir um perfil diferente do [default] perfil como padrão para seu aplicativo, defina o AWS_PROFILE variável de ambiente para o nome do seu perfil personalizado.

Para definir essa variável no Linux, macOS ou Unix, use `export`:

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

Para definir essas variáveis no Windows, use `set`:

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

Como alternativa, defina `oaws .profile` propriedade do sistema Java para o nome do perfil.

Recarregar credenciais do perfil

Você pode configurar qualquer provedor de credenciais que tenha um `profileFile()` método em seu construtor para recarregar as credenciais do perfil. Essas classes de perfil de credenciais são: `ProfileCredentialsProvider`, `DefaultCredentialsProvider`, `InstanceProfileCredentialsProvider` e `ProfileTokenProvider`.

Note

O recarregamento da credencial do perfil funciona somente com as seguintes configurações no arquivo de perfil: `aws_access_key_id`, `aws_secret_access_key`, `eaws_session_token`.

Configurações como `region`, `sso_session`, `sso_account_id`, e `resource_profiles` são ignorados.

Para configurar um provedor de credenciais compatível para recarregar as configurações do perfil, forneça uma instância de `ProfileFileSupplier` para o `profileFile()` método do construtor. O exemplo de código a seguir demonstra um `ProfileCredentialsProvider` que recarrega as configurações de credenciais do [default] perfil.

```
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().
*/
```

Quando `ProfileCredentialsProvider.resolveCredentials()` é chamado, o SDK para Java recarrega as configurações. `ProfileFileSupplier.defaultSupplier()` é um dos [várias implementações de conveniência](#) do `ProfileFileSupplier` fornecido pelo SDK. Se seu caso de uso exigir, você pode fornecer sua própria implementação.

O exemplo a seguir mostra o uso do `ProfileFileSupplier.reloadWhenModified()` método de conveniência. `reloadWhenModified()` leva um `Path` parâmetro, que oferece flexibilidade na designação do arquivo de origem para a configuração em vez do padrão `~/.aws/credentials` (ou `config`) localização.

As configurações serão recarregadas quando `resolveCredentials()` é chamado somente se o SDK determinar que o conteúdo do arquivo foi modificado.

```
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.
*/
```

O `ProfileFileSupplier.aggregate()` método mescla o conteúdo de vários arquivos de configuração. Você decide se um arquivo é recarregado por chamada `parseResolveCredentials()` ou as configurações de um arquivo foram corrigidas no momento em que ele foi lido pela primeira vez.

O exemplo a seguir mostra um `DefaultCredentialsProvider` que mescla as configurações de dois arquivos que contêm configurações de perfil. O SDK recarrega as configurações no arquivo apontado pelo `credentialsFilePath` variável a cada vez `resolveCredentials()` é chamado e as configurações foram alteradas. As configurações do `profileFile` objeto permanece o mesmo.

```
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.
 */
```

Carregar credenciais temporárias de um processo externo

Warning

A seguir, descrevemos um método de obtenção de credenciais temporárias de um processo externo. Isso pode ser potencialmente perigoso, portanto, proceda com cuidado. Outros provedores de credenciais devem ser preferidos, se possível. Se estiver usando essa opção, você deve se certificar de que o arquivo é o mais bloqueado possível usando as melhores práticas de segurança para seu sistema operacional.

Certifique-se de que sua ferramenta de credenciais personalizadas não grava nenhuma informação secreta em `StdErr`. SDKs e AWS CLI podem capturar e registrar essas informações, potencialmente expondo-as a usuários não autorizados.

Com o SDK para Java 2.x, você pode adquirir credenciais temporárias de um processo externo para casos de uso personalizados. Há duas maneiras de configurar essa funcionalidade.

Use o `credential_process` definição

Se você tiver um método que forneça credenciais temporárias, poderá integrá-lo adicionando `credential_process` configuração como parte de uma definição de perfil no config arquivo. O valor especificado deve usar o caminho completo para o arquivo de comando. Se o caminho do arquivo contiver espaços, você deverá colocá-lo entre aspas.

O SDK chama o comando exatamente como indicado e, em seguida, lê os dados JSON de `stdout`.

Os exemplos a seguir mostram o uso dessa configuração para caminhos de arquivo sem espaços e caminhos de arquivo com espaços.

Linux/macOS

Sem espaços no caminho do arquivo

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

Espaços no caminho do arquivo

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

Windows

Sem espaços no caminho do arquivo

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

Espaços no caminho do arquivo

```
[profile process-credential-profile]
```

```
credential_process = "C:\Path\With Space To\credentials.cmd" --custom_command
custom_parameter
```

O trecho de código a seguir demonstra como criar um cliente de serviço que usa as credenciais temporárias definidas como parte do perfil chamado `process-credential-profile`.

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

Para obter informações detalhadas sobre o uso de um processo externo como fonte de credenciais temporárias, consulte [seção de credenciais do processo](#) no AWS Guia de referência de SDKs e ferramentas.

Usar a `ProcessCredentialsProvider`

Como alternativa ao uso de configurações no `config` arquivo, você pode usar o SDK's [ProcessCredentialsProvider](#) para carregar credenciais temporárias usando Java.

Os exemplos a seguir mostram várias versões de como especificar um processo externo usando o `ProcessCredentialsProvider` configurando um cliente de serviço que usa as credenciais temporárias.

Linux/macOS

Sem espaços no caminho do arquivo

```
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();
```

Espaços no caminho do arquivo

```
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

Sem espaços no caminho do arquivo

```
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();
```

Espaços no caminho do arquivo

```
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();
```

Forneça credenciais temporárias em código

Se a cadeia de credenciais padrão ou um provedor ou cadeia de fornecedores específicos ou personalizados não funcionarem para seu aplicativo, você poderá fornecer credenciais temporárias diretamente no código. Estes podem ser [Credenciais da função do IAM](#) como [descrito acima](#) ou credenciais temporárias recuperadas de AWS Security Token Service (AWS STS). Se você recuperou credenciais temporárias usando AWS STS, forneça-os a um AWS service (Serviço da AWS) cliente, conforme mostrado no exemplo de código a seguir.

1. Assuma uma função ligando `StsClient.assumeRole()`.
2. Crie um [`StaticCredentialsProvider`](#) objeto e fornece-lo com o `AwsSessionCredentials` objeto.
3. Configure o construtor do cliente de serviço com o `StaticCredentialsProvider` construa o cliente.

O exemplo a seguir cria um cliente de serviço Amazon S3 usando credenciais temporárias retornadas por AWS STS para uma função assumida pelo IAM.

```
// The AWS IAM Identity Center identity (user) who executes this method does not
have permission to list buckets.

// The identity is configured in the [default] profile.
public static void assumeRole(String roleArn, String roleSessionName) {
    // The IAM role represented by the 'roleArn' parameter can be assumed by
    identities in two different accounts
    // and the role permits the user to only list buckets.

    // The SDK's default credentials provider chain will find the single sign-on
    settings in the [default] profile.
    // The identity configured with the [default] profile needs permission to call
    AssumeRole on the STS service.
    try {
        Credentials tempRoleCredentials;
        try (StsClient stsClient = StsClient.create()) {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                .roleArn(roleArn)
                .roleSessionName(roleSessionName)
                .build();
    }
}
```

```

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        tempRoleCredentials = roleResponse.credentials();
    }
    // Use the following temporary credential items for the S3 client.
    String key = tempRoleCredentials.accessKeyId();
    String secKey = tempRoleCredentials.secretAccessKey();
    String secToken = tempRoleCredentials.sessionToken();

    // List all buckets in the account associated with the assumed role
    // by using the temporary credentials retrieved by invoking
stsClient.assumeRole().
    StaticCredentialsProvider staticCredentialsProvider =
StaticCredentialsProvider.create(
        AwsSessionCredentials.create(key, secKey, secToken));
    try (S3Client s3 = S3Client.builder()
        .credentialsProvider(staticCredentialsProvider)
        .build()) {
        List<Bucket> buckets = s3.listBuckets().buckets();
        for (Bucket bucket : buckets) {
            System.out.println("bucket name: " + bucket.name());
        }
    }
} catch (StsException | S3Exception e) {
    logger.error(e.getMessage());
    System.exit(1);
}
}
}

```

Conjunto de permissões

O seguinte conjunto de permissões definido em AWS IAM Identity Center permite que a identidade (usuário) execute as duas operações a seguir

1. O `GetObject` operação do Amazon Simple Storage Service.
2. O `AssumeRole` operação do AWS Security Token Service.

Sem assumir o papel, o `s3.listBuckets()` método mostrado no exemplo falharia.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {

```

```
"Effect": "Allow",
"Action": [
    "s3:GetObject",
    "sts:AssumeRole"
],
"Resource": [
    "*"
]
}
```

Função assumida

Política de permissões de funções assumidas

A política de permissões a seguir está anexada à função assumida no exemplo anterior. Essa política de permissões permite listar todos os buckets na mesma conta da função.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListAllMyBuckets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Política de confiança da função assumida

A política de confiança a seguir está anexada à função assumida no exemplo anterior. A política permite que a função seja assumida por identidades (usuários) em duas contas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {

```

```
        "Effect": "Allow",
        "Principal": {
            "AWS": [
                "arn:aws:iam::111122223333:root",
                "arn:aws:iam::555555555555:root"
            ]
        },
        "Action": "sts:AssumeRole",
        "Condition": {}
    }
]
```

Configurar IAM funções para Amazon EC2

Você pode usar um perfil do IAM com o objetivo de gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer solicitações da AWS CLI ou da API da AWS. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma função da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém a função e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para obter mais informações, consulte [Uso de um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2](#) no Guia do usuário do IAM.

Este tópico fornece informações sobre como usar as funções do IAM com aplicativos do AWS SDK for Java em execução no Amazon EC2.

Cadeia de fornecedores padrão e Amazon EC2 perfis de instância

Se seu aplicativo criar um AWSClient usando o `create` método, o cliente procura credenciais temporárias usando a cadeia de fornecedores de credenciais padrão, conforme descrito na [seção called “Cadeia de provedores de credenciais padrão”](#).

A etapa final na cadeia de fornecedores padrão está disponível somente durante a execução do aplicativo em uma instância do Amazon EC2. No entanto, ele oferece maior facilidade de uso e segurança ao trabalhar com Amazon EC2 instâncias. Você também pode passar por um [InstanceProfileCredentialsProvider](#) instance diretamente para o construtor do cliente para obter as credenciais do perfil da instância sem passar por toda a cadeia de fornecedores padrão.

Isso é demonstrado no exemplo a seguir:

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

Quando você usa essa abordagem, o SDK recupera temporariamente AWS credenciais que têm as mesmas permissões que as associadas ao IAM papel que está associado ao Amazon EC2 instância em seu perfil de instância. Embora essas credenciais sejam temporárias e acabem expirando, InstanceProfileCredentialsProvider os atualiza periodicamente para você, para que continuem a permitir o acesso a AWS.

Demonstração: usar funções do IAM em instâncias do EC2

Esta demonstração mostra como recuperar um objeto do Amazon S3 usando uma função do IAM para gerenciar acesso.

Criar uma função do IAM

Crie uma IAM função que concede acesso somente de leitura a Amazon S3.

1. Abra o [console do IAM](#).
2. No painel de navegação, escolha Funções, então Criar função.
3. Sobre o Selecione uma entidade confiável página, escolha AWS service (Serviço da AWS). Em Use case (Caso de uso), selecione EC2. Escolha Next (próximo).
4. Sobre o Adicionar permissões página, escolha Amazon S3ReadOnlyAccess na lista de políticas de permissões e, em seguida, escolha Próximo.

Insira um nome para a função e selecione Criar função. Anote esse nome. Você precisa dele para lançar sua Amazon EC2 instância.

Inicie uma instância do EC2 e especifique sua função do IAM

Para iniciar uma instância do Amazon EC2 com uma função do IAM usando o console, siga as instruções em [o Guia do usuário do Amazon EC2 para instâncias Linux](#).

Note

Você precisará criar ou usar um grupo de segurança e par de chaves para se conectar à instância.

Com isso I AM e Amazon EC2 configuração, você pode implantar seu aplicativo no Amazon EC2 instância e ele terá acesso de leitura a Amazon S3.

AWS seleção da região

As regiões permitem que você acesse AWS serviços que residem fisicamente em uma área geográfica específica. Isso pode ser útil para redundância e para manter os dados e os aplicativos em execução próximo ao lugar onde você e os usuários os acessarão.

Dentro AWS SDK for Java 2.x, todas as diferentes classes relacionadas a região da versão 1.x foram recolhidas em uma classe de região. Você pode usar essa classe para todas as ações relacionadas a região, como recuperar metadados sobre uma região ou verificar se um serviço está disponível em uma região.

Escolher uma região

Você pode especificar o nome de uma região, e o SDK escolherá automaticamente um endpoint apropriado.

Para definir explicitamente uma região, recomendamos que você use as constantes definidas na classe [Region](#). Esta é uma enumeração de todas as regiões disponíveis publicamente. Para criar um cliente com uma região da classe, use o código a seguir.

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

Se a região que você está tentando usar não for uma das constantes na classe Region, você poderá criar uma região usando o método of. Esse recurso permite que você acesse as novas regiões sem atualizar o SDK.

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

Note

Depois que você compilar um cliente com o compilador, ele será imutável, e a região não poderá ser alterada. Se você estiver trabalhando com vários Regiões da AWS Para o mesmo serviço, você deve criar vários clientes — um por região.

Escolher um endpoint específico

Each AWS cliente pode ser configurado para usar um endpoint específico dentro de uma região chamando o `endpointOverrideMethod`.

Por exemplo, para configurar o Amazon EC2 Para usar a região da Europa (Irlanda), use o código a seguir.

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

Consulte [Regiões e endpoints](#) do Para a lista atual de regiões e os endpoints correspondentes a todos AWS Serviços da .

Determine automaticamente a região a partir do ambiente

Ao executar no Amazon EC2 ou no AWS Lambda, você pode configurar os clientes para usar a mesma região na qual o código está em execução. Isso desvincula o código do ambiente no qual está em execução e facilita ainda mais a implantação do aplicativo em várias regiões tendo em vista menos latência ou redundância.

Para usar a cadeia de provedores de credencial/região padrão a fim de determinar a região do ambiente, use o método `create` do compilador de cliente.

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

Se você não definir explicitamente uma região usando o método `region`, o SDK consultará a cadeia de fornecedores da região padrão para tentar e determinar a região a ser usada.

Cadeia de provedor de região padrão

Este é o processo de pesquisa da região:

1. Qualquer região explícita definida usando-se `region` no compilador propriamente dito tem precedência sobre todo o resto.
2. A variável de ambiente `AWS_REGION` está marcada. Se estiver definida, essa região será usada para configurar o cliente.

 Note

Essa variável de ambiente é definida pelo contêiner do Lambda.

3. O SDK verifica o AWS arquivo de configuração compartilhado (geralmente localizado em `~/.aws/config`). Se a propriedade da região estiver presente, o SDK a usará.
 - A variável de ambiente `AWS_CONFIG_FILE` pode ser usada para personalizar o local do arquivo de configuração compartilhado.
 - A `AWS_PROFILE` variável de ambiente ou `oaws.profile` A propriedade do sistema pode ser usada para personalizar o perfil que o SDK carrega.
4. O SDK tentar usar o serviço de metadados da instância do Amazon EC2 para determinar a região da instância do Amazon EC2 em execução no momento.
5. Se o SDK ainda não tiver encontrado uma região a esta altura, a criação do cliente falhará com uma exceção.

Durante o desenvolvimento AWS aplicativos, uma abordagem comum é usar o arquivo de configuração compartilhado (descrito em [Ordem de recuperação de credenciais](#)) definir a região para o desenvolvimento local e contar com a cadeia de fornecedores da região padrão para determinar a região durante a execução na AWS Infraestrutura. Isso simplifica muito a criação do cliente e mantém a portabilidade do aplicativo.

Verificando a disponibilidade do serviço em uma região

Para ver se um determinado AWS service (Serviço da AWS) está disponível em uma região, use o `serviceMetadata` e `regionMethod` no serviço que você gostaria de verificar.

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

Consulte a documentação da classe [Region](#) das regiões que você pode especificar, e use o prefixo de endpoint do serviço a ser consultado.

Reduza o tempo de inicialização do SDK para AWS Lambda

Um dos objetivos do AWS SDK for Java 2.x é reduzir a latência de inicialização para AWS Lambda funções. O SDK contém alterações que reduzem o tempo de inicialização, que são discutidas no final deste tópico.

Primeiro, este tópico se concentra nas mudanças que você pode fazer para reduzir os tempos de inicialização a frio. Isso inclui fazer alterações na estrutura do código e na configuração dos clientes de serviço.

Use os SDKs `URLConnectionHttpClient`

Para sincronoscenários, o SDK para Java 2.x oferece a classe [URLConnectionHttpClient](#), que se baseia nas classes de cliente HTTP do JDK. Porque o `URLConnectionHttpClient` é baseado em classes que já estão no classpath, não há dependências extras para carregar.

Para obter informações sobre como adicionar o `URLConnectionHttpClient` para seu projeto Lambda e configurando seu uso, consulte [Configurar o cliente HTTP baseado em URLConnection](#).

Note

Existem algumas limitações de recursos com o `URLConnectionHttpClient` em comparação com os SDKs [ApacheHttpClient](#). O `ApacheHttpClient` é o cliente HTTP assíncrono padrão no SDK. Por exemplo, o `URLConnectionHttpClient` não suporta o método HTTP PATCH.

Um punhado de AWSAs operações de API exigem solicitações de PATCH. Esses nomes de operação geralmente começam com `Update*`. Veja a seguir vários exemplos.

- [Vários `Update*` operações](#) no AWS Security Hub API e também a [BatchUpdateFindings](#) operação
- Todas as APIs do Amazon API Gateway [Update* operações](#)
- [Vários `Update*` operações](#) na Amazônia WorkDocs API

Se você puder usar o `URLConnectionHttpClient`, primeiro consulte a Referência da API para o AWS service (Serviço da AWS) que você está usando. Verifique se as operações necessárias usam a operação PATCH.

Use os SDKs `AwsCrtAsyncHttpClient`

O `AwsCrtAsyncHttpClient` é a assíncronocontrapartida para reduzir o tempo de inicialização do Lambda no SDK.

O `AwsCrtAsyncHttpClient` é um cliente HTTP assíncrono e sem bloqueio. É construído com base nas ligações Java do AWS Common Runtime, escrito na linguagem de programação C. Entre os objetivos no desenvolvimento do AWS Common Runtime é um desempenho rápido.

A seção deste guia sobre [configurando clientes HTTP](#) tem informações sobre como adicionar um `AwsCrtAsyncHttpClient` ao seu projeto Lambda e configurando seu uso.

Remover dependências não utilizadas do cliente HTTP

Junto com o uso explícito de `URLConnectionHttpClient` ou `AwsCrtAsyncHttpClient`, você pode remover outros clientes HTTP que o SDK traz por padrão. O tempo de inicialização do Lambda é reduzido quando menos bibliotecas precisam ser carregadas, então você deve remover quaisquer artefatos não utilizados que a JVM precise carregar.

O seguinte trecho de um Maven `pom.xml` arquivo mostra a exclusão do cliente HTTP baseado em Apache e do cliente HTTP baseado em Netty. (Esses clientes não são necessários quando você usa `URLConnectionHttpClient`.) Este exemplo exclui os artefatos do cliente HTTP da dependência do cliente S3 e adiciona o `url-connection-client` artefato, que traz `URLConnectionHttpClient` classe.

```
<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>
```

Se você usar o `AwsCrtAsyncHttpClient`, substitua a dependência `url-connection-client` uma dependência do `aws-crt-client`.

 Note

Adicione o `<exclusions>` elemento para todas as dependências do cliente de serviço em seu `pom.xml` arquivo.

Configurar clientes de serviço para pesquisar atalhos

Especifique uma região

Ao criar um cliente de serviço, ligue para o `region` método no construtor de clientes de serviço. Isso reduz o atalho padrão do SDK [Processo de pesquisa de região](#) que verifica vários lugares para o `Region` da AWS informado.

Para manter o código Lambda independente da região, use o código a seguir dentro do `region` método. Esse código acessa o `AWS_REGION` variável de ambiente definida pelo contêiner Lambda.

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

Usar a `EnvironmentVariableCredentialProvider`

Assim como o comportamento padrão de pesquisa das informações da região, o SDK procura credenciais em vários lugares. Ao especificar o [EnvironmentVariableCredentialProvider](#) ao criar um cliente de serviço, você economiza tempo no processo de pesquisa do SDK.

 Note

O uso desse provedor de credenciais permite que o código seja usado em Lambda funções, mas pode não funcionar em Amazon EC2 ou outros sistemas.

O trecho de código a seguir mostra um cliente de serviço S3 configurado adequadamente para uso em um ambiente Lambda.

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

Initialize o cliente SDK fora do manipulador de funções do Lambda

Recomendamos inicializar um cliente SDK fora do método manipulador Lambda. Dessa forma, se o contexto de execução for reutilizado, a inicialização do cliente de serviço poderá ser ignorada. Ao reutilizar a instância do cliente e suas conexões, as invocações subsequentes do método manipulador ocorrem mais rapidamente.

No exemplo a seguir, o S3Client instância é inicializada no construtor usando um método estático de fábrica. Se o contêiner gerenciado pelo ambiente Lambda for reutilizado, o S3Client instância é reutilizada.

```
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 a injeção de dependência

As estruturas de injeção de dependência (DI) podem levar mais tempo para concluir o processo de configuração. Eles também podem exigir dependências adicionais, que levam tempo para serem carregadas.

Se uma estrutura de DI for necessária, recomendamos o uso de estruturas de DI leves, como [Adaga](#).

Use uma mira do Maven Archetype AWS Lambda

A AWS equipes do Java SDK desenvolveu um [Arquétipo Maven](#) modelo para inicializar um projeto Lambda com tempo mínimo de inicialização. Você pode criar um projeto Maven a partir do arquétipo e saber que as dependências estão configuradas adequadamente para o ambiente Lambda.

Para saber mais sobre o arquétipo e trabalhar com um exemplo de implantação, consulte [postagem no blog](#).

Considere o LambdaSnapStartpara Java

Se seus requisitos de tempo de execução forem compatíveis,AWSofertas[LambdaSnapStartpara Java](#). LambdaSnapStarté uma solução baseada em infraestrutura que melhora o desempenho de inicialização das funções Java. Quando você publica uma nova versão de uma função, o LambdaSnapStartinicializa-o e tira um instantâneo imutável e criptografado da memória e do estado do disco. SnapStartem seguida, armazena o instantâneo em cache para reutilização.

Alterações na versão 2.x que afetam o tempo de inicialização

Além das alterações que você faz no seu código, a versão 2.x do SDK para Java inclui três alterações principais que reduzem o tempo de inicialização:

- Uso de[jackson-jr](#), que é uma biblioteca de serialização que melhora o tempo de inicialização
- Uso do[java.time](#)bibliotecas para objetos de data e hora, que fazem parte do JDK
- Uso de[SLF4j](#)para uma fachada madeireira

Recursos adicionais

O AWS LambdaO Guia do Desenvolvedor contém um[seção sobre melhores práticas](#)para desenvolver funções do Lambda que não sejam específicas de Java.

Por exemplo, como criar um aplicativo nativo da nuvem em Java que usaAWS Lambda, veja isso[conteúdo do workshop](#). O workshop discute a otimização do desempenho e outras melhores práticas.

Você pode considerar o uso de imagens estáticas que são compiladas com antecedência para reduzir a latência de inicialização. Por exemplo, você pode usar o SDK para Java 2.x e o Maven para[construir uma imagem nativa do GraalVM](#).

Clientes HTTP

Você pode alterar a configuração padrão para clientes HTTP em aplicativos que você cria com oAWS SDK for Java 2.x. Esta seção discute os clientes HTTP e as configurações do SDK.

Cientes disponíveis no SDK para Java

Cientes síncronos

Um cliente de serviço síncrono, como o[Cliente S3](#) ou o[DynamoDbClient](#), requer o uso de um cliente HTTP síncrono. O AWS SDK for Java oferece dois clientes HTTP síncronos.

ApacheHttpClient(padrão)

[ApacheHttpClient](#) é o cliente HTTP padrão para clientes de serviços síncronos. Para obter informações sobre como configurar oApacheHttpClient, veja[Configurar o cliente HTTP baseado em Apache](#).

URLConnectionHttpClient

Como uma opção mais leve para oApacheHttpClient, você pode usar o[URLConnectionHttpClient](#). Para obter informações sobre como configurar oURLConnectionHttpClient, veja[Configurar o cliente HTTP baseado em URLConnection](#).

Cientes assíncronos

Um cliente de serviço assíncrono, como o[S3AsyncClient](#) ou o[DynamoDbAsyncClient](#), requer o uso de um cliente HTTP assíncrono. O AWS SDK for Java oferece dois clientes HTTP assíncronos.

NettyNioAsyncHttpClient(padrão)

[NettyNioAsyncHttpClient](#) é o cliente HTTP padrão usado por clientes assíncronos. Para obter informações sobre como configurar oNettyNioAsyncHttpClient, veja[the section called “Configurar o cliente HTTP baseado em Netty”](#).

AwsCrtAsyncHttpClient

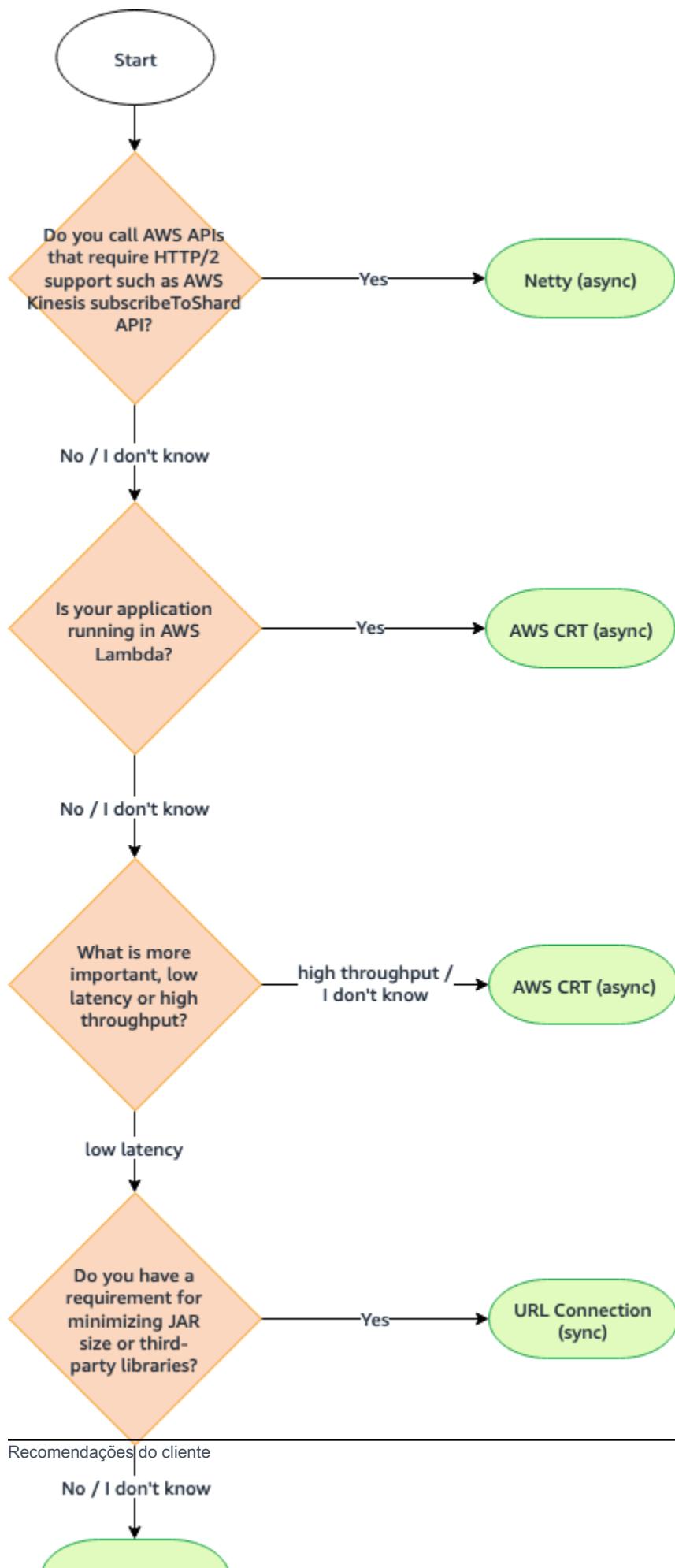
O novo[AwsCrtAsyncHttpClient](#), que também tem um tempo de carregamento mais rápido em comparação com oNettyNioAsyncHttpClient, também está disponível. Para obter informações sobre como configurar oAwsCrtAsyncHttpClient, veja[the section called “Configurar o cliente AWS HTTP baseado em CRT”](#).

Recomendações do cliente HTTP

Vários fatores entram em jogo quando você escolhe uma implementação de cliente HTTP. Use as informações a seguir para ajudá-lo a decidir.

Fluxograma de recomendação

O fluxograma a seguir fornece orientação geral para ajudá-lo a determinar qual cliente HTTP usar.



Comparação de clientes HTTP

A tabela a seguir fornece informações detalhadas para cada cliente HTTP.

Cliente HTTP	Sincronizado ou assíncrono	Quando usar	Limitação/ desvantagem
Cliente HTTP baseado em Apache (cliente HTTP de sincronização padrão)	Sincronização	Use-o se você preferir baixa latência em vez de alta taxa de transferência	Tempo de inicialização mais lento em comparação com outros clientes HTTP
Cliente HTTP baseado em conexão de URL	Sincronização	Use-o se você tiver um requisito difícil para limitar dependências de terceiros	Não é compatível com o método HTTP PATCH, exigido por algumas APIs, como as operações de atualização do Amazon ApiGateway
Cliente HTTP baseado em Netty (cliente HTTP assíncrono padrão)	Assíncrono	<ul style="list-style-type: none"> Use-o se seu aplicativo invocar APIs que exijam suporte a HTTP/2, como a API KinesisSubscribe ToShard 	Tempo de inicialização mais lento em comparação com outros clientes HTTP
AWSCliente HTTP baseado em CRT ¹	Assíncrono	<ul style="list-style-type: none"> Use-o se seu aplicativo estiver sendo executado em AWS Lambda Use-o se você preferir alto rendimento em vez de baixa latência 	<ul style="list-style-type: none"> Não oferece suporte a clientes de serviços que

Cliente HTTP	Sincronizado ou assíncrono	Quando usar	Limitação/ desvantagem
			exigem suporte HTTP/2, como KinesisAsyncClient e TranscribeStreamingAsyncClient

¹ Recomendamos que você use o AWS Cliente HTTP baseado em CRT, se possível, por causa de seus benefícios adicionais.

Padrões de configuração inteligente

O AWS SDK for Java 2.x (versão 2.17.102 ou posterior) oferece um recurso de padrões de configuração inteligente. Esse recurso otimiza duas propriedades do cliente HTTP junto com outras propriedades que não afetam o cliente HTTP.

Os padrões de configuração inteligente definem valores sensatos para o connectTimeoutInMillis e negotiationTimeoutInMillis propriedades com base em um valor de modo padrão que você fornece. Você escolhe o valor do modo padrão com base nas características do seu aplicativo.

Para obter mais informações sobre os padrões de configuração inteligente e como escolher o valor do modo padrão mais adequado para seus aplicativos, consulte o [AWS Guia de referência de SDKs e ferramentas](#).

A seguir estão quatro maneiras de definir o modo padrão para seu aplicativo.

Service client

Use o construtor do cliente de serviço para configurar o modo padrão diretamente no cliente de serviço. O exemplo a seguir define o modo padrão como auto para o DynamoDbClient.

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

```
.build();
```

System property

Você pode usar `oaws.defaultsMode` propriedade do sistema para especificar o modo padrão. Se você definir a propriedade do sistema em Java, precisará definir a propriedade antes de inicializar qualquer cliente de serviço.

O exemplo a seguir mostra como definir o modo padrão como auto usando uma propriedade do sistema definida em Java.

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

O exemplo a seguir demonstra como você define o modo padrão como auto usando um -Dopção do javacomando.

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

Environment variable

Defina um valor para a variável de ambiente `AWS_DEFAULTS_MODE` para selecionar o modo padrão para seu aplicativo.

As informações a seguir mostram o comando a ser executado para definir o valor do modo padrão como auto usando uma variável de ambiente.

Sistema operacional	Comando para definir variáveis de ambiente
Linux, macOS ou Unix	<code>export AWS_DEFAULTS_MODE=auto</code>
Windows	<code>set AWS_DEFAULTS_MODE=auto</code>

AWS config file

Você pode adicionar um `defaults_mode` propriedade de configuração para o compartilhado `AWS config` arquivo como mostra o exemplo a seguir.

```
[default]
```

```
defaults_mode = auto
```

Se você definir o modo padrão globalmente com a propriedade do sistema, a variável de ambiente ou AWSarquivo de configuração, você pode substituir as configurações ao criar um cliente HTTP.

Quando você cria um cliente HTTP com `oHttpClientBuilder()` método, as configurações se aplicam somente à instância que você está criando. Um exemplo disso é mostrado[aqui](#). O cliente HTTP baseado em Netty neste exemplo substitui todos os valores de modo padrão definidos globalmente para `connectTimeoutInMillis`, `setTlsNegotiationTimeoutInMillis`.

Suporte de proxy

Você pode configurar proxies HTTP usando código, definindo propriedades do sistema Java ou combinando as duas abordagens. Atualmente, o SDK não oferece suporte a variáveis de ambiente para configurar proxies.

Você configura proxies em código com um cliente específico `ProxyConfiguration` construtor quando você cria o cliente de serviço. A seção de cada cliente HTTP neste tópico mostra um exemplo de configuração de proxy. Isso[exemplo é para o cliente Apache HTTP](#).

Suporte ao cliente HTTP das propriedades do sistema Java para proxies HTTP

Propriedade do sistema	Descrição	Suporte ao cliente HTTP
<code>Http.ProxyHost</code>	Nome do host do servidor proxy HTTP	Tudo
<code>Porta http.proxy</code>	Número da porta do servidor proxy HTTP	Tudo
<code>Usuário http.proxy</code>	Nome de usuário para autenticação de proxy HTTP	Tudo
<code>Senha http.proxy</code>	Senha para autenticação de proxy HTTP	Tudo
<code>http.nonProxyHosts</code>	Lista de hosts que devem ser acessados diretamente, ignorando o proxy. Também	Tudo

Propriedade do sistema	Descrição	Suporte ao cliente HTTP
	<u>é válido quando HTTPS é usado.</u>	
https://ProxyHost	Nome do host do servidor proxy HTTPS	Netty, CRT
HTTPS.ProxyPort	Número da porta do servidor proxy HTTPS	Netty, CRT
Usuário https. Proxy	Nome de usuário para autenticação de proxy HTTPS	Netty, CRT
HTTPS.proxy Senha	Senha para autenticação de proxy HTTPS	Netty, CRT

Os termos usados nas tabelas significam:

- Todos: todos os clientes HTTP oferecidos pelo SDK
—`URLConnectionHttpClient`, `ApacheHttpClient`, `NettyNioAsyncHttpClient`, `AwsCrtAsyncHttp`
- Netty: o cliente HTTP baseado em Netty (`NettyNioAsyncHttpClient`)
- CRT: o AWSClient HTTP baseado em CRT, (`AwsCrtAsyncHttpClient`)

Você pode usar uma combinação de configuração do cliente HTTP e propriedades do sistema. Cada cliente `HTTPProxyConfiguration` construtor fornece `useSystemPropertyValues` configuração. Por padrão, a configuração é `true`. Quando a configuração é `true`, o SDK usa automaticamente os valores das propriedades do sistema para opções que não são fornecidas usando o `ProxyConfiguration` construtor.

O exemplo a seguir mostra a configuração fornecida por uma propriedade do sistema e pelo código.

```
// Command line with the proxy password set as a system property.
$ java -Dhttp.proxyPassword=password -cp ... App

// Since the 'useSystemPropertyValues' setting is 'true' (the default), the SDK will
// supplement
// the proxy configuration in code with the 'http.proxyPassword' value from the system
// property.
```

```
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://localhost:1234"))
        .username("username")
        .build())
    .build();

// Use the apache HTTP client with proxy configuration.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(apacheHttpClient)
    .build();
```

Note

Em vez de definir o `endpoint` propriedade no código conforme mostrado no trecho de código anterior, você pode usar as seguintes propriedades do sistema.

```
-Dhttp.proxyHost=localhost -Dhttp.proxyPort=1234
```

Configurar o cliente HTTP baseado em Apache

Clientes de serviço síncrono no AWS SDK for Java 2.x usam um cliente HTTP baseado em Apache, por padrão. [ApacheHttpClient](#) Os SDKs ApacheHttpClient são baseados no [HttpClient](#) Apache.

O SDK também oferece o [URLConnectionHttpClient](#), que carrega mais rapidamente, mas tem menos recursos. Para obter informações sobre como configurar o `URLConnectionHttpClient`, consulte [the section called “Configurar o cliente HTTP baseado em `URLConnection`”](#).

[Para ver o conjunto completo de opções de configuração disponíveis para você ApacheHttpClient, consulte ApacheHttpClient.Builder e ProxyConfiguration.Builder.](#)

Acesse o [ApacheHttpClient](#)

Na maioria das situações, você usa o ApacheHttpClient sem nenhuma configuração explícita. Você declara seus clientes de serviço e o SDK os configurará ApacheHttpClient com valores padrão para você.

Se você quiser configurá-lo explicitamente ApacheHttpClient ou usá-lo com vários clientes de serviço, você precisa disponibilizá-lo para configuração.

Nenhuma configuração é necessária

Quando você declara uma dependência de um cliente de serviço no Maven, o SDK adiciona uma dependência de tempo de execução ao artefato. apache-client Isso torna a ApacheHttpClient classe disponível para seu código em tempo de execução. Se você não estiver configurando o cliente HTTP baseado em Apache, não precisará especificar uma dependência para ele.

No seguinte trecho XML de um pom.xml arquivo Maven, a dependência declarada com traz <artifactId>s3</artifactId> automaticamente o cliente HTTP baseado em Apache. Você não precisa declarar uma dependência especificamente para isso.

```
<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>
```

Com essas dependências, você não pode fazer nenhuma alteração explícita na configuração HTTP, porque a ApacheHttpClient biblioteca está somente no caminho de classe do tempo de execução.

Configuração necessária

Para configurar oApacheHttpClient, você precisa adicionar uma dependência na apache-client biblioteca em tempo de compilação.

Consulte o exemplo a seguir de um pom.xml arquivo Maven para configurar o 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>
```

Configurar o **ApacheHttpClient**

Você pode configurar uma instância do `ApacheHttpClient` junto com a criação de um cliente de serviço ou configurar uma única instância para compartilhar entre vários clientes de serviço.

Com qualquer uma das abordagens, você usa o [ApacheHttpClient.Builder](#) para configurar as propriedades do cliente HTTP baseado em Apache.

Melhor prática: dedicar uma **ApacheHttpClient** instância a um cliente de serviço

Se você precisar configurar uma instância do `ApacheHttpClient`, recomendamos que você crie a `ApacheHttpClient` instância dedicada. Você pode fazer isso usando o `httpClientBuilder` método do construtor do cliente de serviço. Dessa forma, o ciclo de vida do cliente HTTP é gerenciado pelo SDK, o que ajuda a evitar possíveis vazamentos de memória se a `ApacheHttpClient` instância não for fechada quando não for mais necessária.

O exemplo a seguir cria um `S3Client` e configura a instância incorporada de `ApacheHttpClient` with `maxConnections` e `connectionTimeout` values. A instância HTTP é criada usando o `httpClientBuilder` método de `S3Client.Builder`.

Importações

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

Código

```
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.
```

Abordagem alternativa: compartilhar uma `ApacheHttpClient` instância

Para ajudar a reduzir o uso de recursos e memória em seu aplicativo, você pode configurar um `ApacheHttpClient` e compartilhá-lo entre vários clientes de serviço. O pool de conexões HTTP será compartilhado, o que reduz o uso de recursos.

Note

Quando uma `ApacheHttpClient` instância é compartilhada, você deve fechá-la quando ela estiver pronta para ser descartada. O SDK não fechará a instância quando o cliente de serviço for fechado.

O exemplo a seguir configura um cliente HTTP baseado em Apache, que é usado por dois clientes de serviço. A `ApacheHttpClient` instância configurada é passada para o `httpClient` método do construtor do cliente de serviço. Quando os clientes de serviço e o cliente HTTP não são mais necessários, eles são explicitamente fechados. O cliente HTTP é fechado por último.

Importações

```
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;
```

Código

```
SdkHttpClient apacheHttpClient = ApacheHttpClient.create();

// 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.
```

Exemplo de configuração de proxy

O trecho de código a seguir usa o [construtor de configuração de proxy para o cliente HTTP Apache](#).

```
SdkHttpClient apacheHttpClient = ApacheHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://example.com:1234"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .addNonProxyHost("host.example.com")
        .build())
    .build();
```

As propriedades equivalentes do sistema Java para a configuração do proxy são mostradas no seguinte trecho de linha de comando.

```
$ java -Dhttp.proxyHost=example.com -Dhttp.proxyPort=1234 -Dhttp.proxyUser=username \
-Dhttp.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...
App
```

 Note

Atualmente, o cliente Apache HTTP não oferece suporte às propriedades do sistema proxy HTTPS.

Configurar o cliente HTTP baseado em URLConnection

O AWS SDK for Java 2.x oferece um cliente [URLConnectionHttpClient](#) HTTP mais leve em comparação com o padrão ApacheHttpClient. O [URLConnectionHttpClient](#) é baseado em Java[URLConnection](#).

Ele [URLConnectionHttpClient](#) carrega mais rapidamente do que o cliente HTTP baseado em Apache, mas tem menos recursos. Como carrega mais rapidamente, é uma [boa solução](#) para AWS Lambda funções Java.

O [URLConnectionHttpClient](#) tem várias [opções configuráveis](#) que você pode acessar.

Para saber como configurar o cliente HTTP baseado em Apache, consulte [Configurar o cliente HTTP baseado em Apache](#)

 Note

O [URLConnectionHttpClient](#) não suporta o método HTTP PATCH.

Algumas operações de AWS API exigem solicitações de PATCH. Esses nomes de operação geralmente começam comUpdate*. Veja a seguir alguns exemplos.

- [Várias Update* operações](#) na AWS Security Hub API e também a [BatchUpdateFindings](#) operação
- Todas as [Update*operações](#) de API do Amazon API Gateway
- [Várias Update* operações](#) na WorkDocs API da Amazon

Se você puder usar o `URLConnectionHttpClient`, primeiro consulte a referência da API AWS service (Serviço da AWS) que você está usando. Verifique se as operações necessárias usam a operação PATCH.

Acesse o `URLConnectionHttpClient`

Para configurar e usar o `URLConnectionHttpClient`, você declara uma dependência do artefato `url-connection-client` Maven em seu arquivo. `pom.xml`

Ao contrário do `ApacheHttpClient`, o não `URLConnectionHttpClient` é adicionado automaticamente ao seu projeto, portanto, o uso deve declará-lo especificamente.

O exemplo de `pom.xml` arquivo a seguir mostra as dependências necessárias para usar e configurar o cliente HTTP.

```
<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>
```

Configurar o `URLConnectionHttpClient`

Você pode configurar uma instância do `URLConnectionHttpClient` junto com a criação de um cliente de serviço ou configurar uma única instância para compartilhar entre vários clientes de serviço.

Com qualquer uma das abordagens, você usa o `URLConnectionHttpClient.Builder` para configurar as propriedades do cliente HTTP baseado em `URLConnection`.

Melhor prática: dedicar uma `URLConnectionHttpClient` instância a um cliente de serviço

Se você precisar configurar uma instância do `URLConnectionHttpClient`, recomendamos que você crie a `URLConnectionHttpClient` instância dedicada. Você pode fazer isso usando o `httpClientBuilder` método do construtor do cliente de serviço. Dessa forma, o ciclo de vida do cliente HTTP é gerenciado pelo SDK, o que ajuda a evitar possíveis vazamentos de memória se a `URLConnectionHttpClient` instância não for fechada quando não for mais necessária.

O exemplo a seguir cria um `S3Client` e configura a instância incorporada de `URLConnectionHttpClient` with `maxConnections` e `connectionTimeout` values. A instância HTTP é criada usando o `httpClientBuilder` método de `S3Client.Builder`.

O exemplo a seguir cria um `S3Client` e configura a instância incorporada de `URLConnectionHttpClient` with `socketTimeout` e `proxyConfiguration` values. O `proxyConfiguration` método usa uma expressão lambda Java do tipo `Consumer<ProxyConfiguration.Builder>`. A instância HTTP é criada usando o `httpClientBuilder` método de `S3Client.Builder`.

Importações

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

Código

```
// 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.
```

Abordagem alternativa: compartilhar uma **URLConnectionHttpClient** instância

Para ajudar a reduzir o uso de recursos e memória em seu aplicativo, você pode configurar um **URLConnectionHttpClient** e compartilhá-lo entre vários clientes de serviço. O pool de conexões HTTP será compartilhado, o que reduz o uso de recursos.

Note

Quando uma **URLConnectionHttpClient** instância é compartilhada, você deve fechá-la quando ela estiver pronta para ser descartada. O SDK não fechará a instância quando o cliente de serviço for fechado.

O exemplo a seguir configura um cliente HTTP baseado em **URLConnection**, que é usado por dois clientes de serviço. A **URLConnectionHttpClient** instância configurada é passada para o **httpClient** método do construtor do cliente de serviço. Quando os clientes de serviço e o cliente HTTP não são mais necessários, eles são explicitamente fechados. O cliente HTTP é fechado por último.

Importações

```
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;
```

Código

```
SdkHttpClient urlHttpClient = UrlConnectionHttpClient.create();

// 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();
```

Use **URLConnectionHttpClient** e **ApacheHttpClient** em conjunto

Ao usar o `URLConnectionHttpClient` em seu aplicativo, você deve fornecer a cada cliente de serviço uma `URLConnectionHttpClient` instância ou uma `ApacheHttpClient` instância usando o `httpClientBuilder` método do construtor do cliente de serviço.

Uma exceção ocorre se seu programa usa vários clientes de serviço e as duas afirmações a seguir são verdadeiras:

- Um cliente de serviço está configurado para usar uma `URLConnectionHttpClient` instância
- Outro cliente de serviço usa o padrão `ApacheHttpClient` sem construí-lo explicitamente com os métodos ou `httpClient()` `httpClientBuilder()`

A exceção indicará que várias implementações HTTP foram encontradas no caminho de classe.

O exemplo de trecho de código a seguir leva a uma exceção.

```
// 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.create();

// Perform work with the s3Client and dynamoDbClient.

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

Evite a exceção configurando explicitamente o S3Client com um 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

Para criar explicitamente o ApacheHttpClient, você deve [adicionar uma dependência](#) do apache-client artefato em seu arquivo de projeto Maven.

Exemplo de configuração de proxy

O trecho de código a seguir usa o [construtor de configuração de proxy para o cliente HTTP de conexão URL](#).

```
SdkHttpClient urlHttpClient = UrlConnectionHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .endpoint(URI.create("http://example.com:1234"))
        .username("username")
        .password("password")
        .addNonProxyHost("localhost")
        .addNonProxyHost("host.example.com")
        .build())
    .build();
```

As propriedades equivalentes do sistema Java para a configuração do proxy são mostradas no seguinte trecho de linha de comando.

```
$ java -Dhttp.proxyHost=example.com -Dhttp.proxyPort=1234 -Dhttp.proxyUser=username \
-Dhttp.proxyPassword=password -Dhttp.nonProxyHosts=localhost|host.example.com -cp ...
App
```

 Note

Atualmente, o cliente HTTP de conexão URL não oferece suporte às propriedades do sistema proxy HTTPS.

Configurar o cliente HTTP baseado em Netty

O cliente HTTP padrão para operações assíncronas no AWS SDK for Java 2.x é baseado em Netty.

[NettyNioAsyncHttpClient](#) O cliente baseado em Netty é baseado na estrutura de rede assíncrona orientada por eventos do projeto Netty.

Como alternativa, você pode usar o novo cliente [HTTP AWS baseado em CRT](#). Este tópico mostra como configurar NettyNioAsyncHttpClient o.

Acesse o **NettyNioAsyncHttpClient**

Na maioria das situações, você usa o NettyNioAsyncHttpClient sem nenhuma configuração explícita em programas assíncronos. Você declara seus clientes de serviço assíncronos e o SDK os configurará NettyNioAsyncHttpClient com valores padrão para você.

Se você quiser configurá-lo explicitamente NettyNioAsyncHttpClient ou usá-lo com vários clientes de serviço, você precisa disponibilizá-lo para configuração.

Nenhuma configuração é necessária

Quando você declara uma dependência de um cliente de serviço no Maven, o SDK adiciona uma dependência de tempo de execução ao artefato. `netty-nio-client` Isso torna a `NettyNioAsyncHttpClient` classe disponível para seu código em tempo de execução. Se você não estiver configurando o cliente HTTP baseado em Netty, não precisará especificar uma dependência para ele.

No seguinte trecho XML de um `pom.xml` arquivo Maven, a dependência declarada com `<artifactId>dynamodb-enhanced</artifactId>` transitivamente traz o cliente HTTP baseado em Netty. Você não precisa declarar uma dependência especificamente para isso.

```
<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>
```

Com essas dependências, você não pode fazer nenhuma alteração na configuração HTTP, pois a `NettyNioAsyncHttpClient` biblioteca está somente no caminho de classe do tempo de execução.

Configuração necessária

Para configurar o `NettyNioAsyncHttpClient`, você precisa adicionar uma dependência do `netty-nio-client` artefato em tempo de compilação.

Consulte o exemplo a seguir de um `pom.xml` arquivo Maven para configurar o `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>
```

Configurar o **NettyNioAsyncHttpClient**

Você pode configurar uma instância do **NettyNioAsyncHttpClient** junto com a criação de um cliente de serviço ou configurar uma única instância para compartilhar entre vários clientes de serviço.

Com qualquer uma das abordagens, você usa o [NettyNioAsyncHttpClient.Builder](#) para configurar as propriedades da instância do cliente HTTP baseada em Netty.

Melhor prática: dedicar uma **NettyNioAsyncHttpClient** instância a um cliente de serviço

Se você precisar configurar uma instância do **NettyNioAsyncHttpClient**, recomendamos que você crie a **NettyNioAsyncHttpClient** instância dedicada. Você pode fazer isso usando o `httpClientBuilder` método do construtor do cliente de serviço. Dessa forma, o ciclo de vida do cliente HTTP é gerenciado pelo SDK, o que ajuda a evitar possíveis vazamentos de memória se a **NettyNioAsyncHttpClient** instância não for fechada quando não for mais necessária.

O exemplo a seguir cria uma `DynamoDbAsyncClient` instância, que também é usada por uma `DynamoDbEnhancedAsyncClient` instância. A `DynamoDbAsyncClient` instância contém a `NettyNioAsyncHttpClient` instância com `connectionTimeout` e `maxConcurrency` valores. A instância HTTP é criada usando o `httpClientBuilder` método de `DynamoDbAsyncClient.Builder`.

Importações

```
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;
```

Código

```
// DynamoDbAsyncClient is the lower-level client used by the enhanced client.
DynamoDbAsyncClient dynamoDbAsyncClient =
    DynamoDbAsyncClient
        .builder()
        .httpClientBuilder(NettyNioAsyncHttpClient.builder())
        .connectionTimeout(Duration.ofMillis(5_000))
        .maxConcurrency(100)
        .tlsNegotiationTimeout(Duration.ofMillis(3_500))
        .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();
```

Abordagem alternativa: compartilhar uma **NettyNioAsyncHttpClient** instância

Para ajudar a reduzir o uso de recursos e memória em seu aplicativo, você pode configurar um **NettyNioAsyncHttpClient** e compartilhá-lo entre vários clientes de serviço. O pool de conexões HTTP será compartilhado, o que reduz o uso de recursos.

Note

Quando uma **NettyNioAsyncHttpClient** instância é compartilhada, você deve fechá-la quando ela estiver pronta para ser descartada. O SDK não fechará a instância quando o cliente de serviço for fechado.

O exemplo a seguir configura um cliente HTTP baseado em Netty, que é usado por dois clientes de serviço. A **NettyNioAsyncHttpClient** instância configurada é passada para o `httpClient` método do construtor de cada cliente de serviço. Quando os clientes de serviço e o cliente HTTP não são mais necessários, eles são explicitamente fechados. O cliente HTTP é fechado por último.

Importações

```
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;
```

Código

```
// Create a NettyNioAsyncHttpClient shared instance.
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.create();

// 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.
```

Exemplo de configuração de proxy

O trecho de código a seguir usa o [construtor de configuração de proxy para o cliente HTTP Netty](#).

```
SdkAsyncHttpClient nettyHttpClient = NettyNioAsyncHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .build())
    .build();
```

As propriedades equivalentes do sistema Java para a configuração do proxy são mostradas no seguinte trecho de linha de comando.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
-Dhttps.proxyPassword=password -cp ... App
```

Important

Para usar qualquer uma das propriedades do sistema proxy HTTPS, a `scheme` propriedade deve ser definida em código para `https`. Se a propriedade do esquema não estiver definida no código, o esquema assumirá como padrão HTTP e o SDK `http.*` procurará somente as propriedades do sistema.

Configurar o cliente AWS HTTP baseado em CRT

O [`AwsCrtAsyncHttpClient`](#) é um novo cliente HTTP assíncrono que você pode usar com o AWS SDK for Java 2.x `AwsCrtAsyncHttpClient`. Isso traz os seguintes benefícios para um cliente HTTP no SDK.

- tempo de inicialização mais rápido do SDK
- menor consumo de memória
- tempo de latência reduzido
- gerenciamento de integridade de conexão
- Balanceamento de carga de DNS

Este tópico é sobre como usar e configurar o cliente HTTP AWS baseado em CRT. [Para obter informações sobre como configurar o cliente HTTP baseado em Netty, consulte o tópico anterior.](#)

AWSComponentes baseados em CRT no SDK

O cliente HTTP AWS baseado em CRT, descrito neste tópico, e o cliente S3 AWS baseado em CRT são componentes diferentes no SDK.

O cliente HTTP AWS baseado em CRT é uma implementação da [`SdkAsyncHttpClient`](#) interface e é usado para comunicação HTTP geral. É uma alternativa à implementação da `SdkAsyncHttpClient` interface pelo Netty e oferece vários benefícios.

O [`cliente S3 AWS baseado em CRT`](#) é uma implementação da `AsyncClient` interface [`S3`](#) e é usado para trabalhar com o serviço Amazon S3. É uma alternativa à implementação da `S3AsyncClient` interface baseada em Java e oferece várias vantagens.

Embora ambos os componentes usem bibliotecas do [`AWSCommon Runtime`](#), o cliente HTTP AWS baseado em CRT não usa a [`biblioteca aws-c-s 3`](#) e não oferece suporte aos recursos da API de

upload de [várias partes do S3](#). O cliente S3 AWS baseado em CRT, por outro lado, foi desenvolvido especificamente para oferecer suporte aos recursos da API de upload de várias partes do S3.

Acesse o **AwsCrtAsyncHttpClient**

Antes de usar o cliente HTTP AWS baseado em CRT, adicione o `aws-crt-client` artefato às dependências do seu projeto.

O Maven a seguir `pom.xml` mostra o cliente HTTP AWS baseado em CRT declarado usando o mecanismo de lista de materiais (BOM).

```
<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>
```

Visite o repositório central do Maven para obter o valor mais recente da [VERSÃO](#).

Configurar o **AwsCrtAsyncHttpClient**

Você pode configurar uma instância do `AwsCrtAsyncHttpClient` junto com a criação de um cliente de serviço ou configurar uma única instância para compartilhar entre vários clientes de serviço.

Com qualquer uma das abordagens, você usa o [AwsCrtAsyncHttpClient.Builder](#) para configurar as propriedades da instância do cliente HTTP AWS baseada em CRT.

Melhor prática: dedicar uma **AwsCrtAsyncHttpClient** instância a um cliente de serviço

Se você precisar configurar uma instância do `AwsCrtAsyncHttpClient`, recomendamos que você crie a `AwsCrtAsyncHttpClient` instância dedicada. Você pode fazer isso usando o `httpClientBuilder` método do construtor do cliente de serviço. Dessa forma, o ciclo de vida do cliente HTTP é gerenciado pelo SDK, o que ajuda a evitar possíveis vazamentos de memória se a `AwsCrtAsyncHttpClient` instância não for fechada quando não for mais necessária.

O exemplo a seguir cria um `S3Client` e configura os `maxConcurrency` valores `AwsCrtAsyncHttpClient` com `connectionTimeout` e.

Importações

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

Código

```
// 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();
```

Abordagem alternativa: compartilhar uma **AwsCrtAsyncHttpClient** instância

Para ajudar a reduzir o uso de recursos e memória em seu aplicativo, você pode configurar um `AwsCrtAsyncHttpClient` e compartilhá-lo entre vários clientes de serviço. O pool de conexões HTTP será compartilhado, o que reduz o uso de recursos.

Note

Quando uma `AwsCrtAsyncHttpClient` instância é compartilhada, você deve fechá-la quando ela estiver pronta para ser descartada. O SDK não fechará a instância quando o cliente de serviço for fechado.

O exemplo a seguir configura uma instância de cliente HTTP AWS baseada em CRT com `connectionTimeout` e valores `maxConcurrency`. A `AwsCrtAsyncHttpClient` instância configurada é passada para o `httpClient` método do construtor de cada cliente de serviço. Quando os clientes de serviço e o cliente HTTP não são mais necessários, eles são explicitamente fechados. O cliente HTTP é fechado por último.

Importações

```
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;
```

Código

```
// Create an AwsCrtAsyncHttpClient shared instance.
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.create();

// 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.
```

Defina o cliente HTTP AWS baseado em CRT como padrão

Para operações assíncronas no AWS SDK for Java 2.x, você pode substituir o `NettyNioAsyncHttpClient` como cliente HTTP assíncrono padrão em seu programa pelo `AwsCrtAsyncHttpClient`.

Você define isso no `pom.xml` arquivo Maven do seu projeto excluindo a dependência do `netty-nio-client` para cada cliente de serviço. Como alternativa, você pode definir o cliente HTTP padrão usando uma propriedade do sistema Java ao executar o aplicativo ou no código do aplicativo.

Remova o Netty das dependências do projeto

O `pom.xml` exemplo a seguir remove o cliente HTTP baseado em Netty do caminho de classe para que o cliente HTTP AWS baseado em CRT seja usado em seu lugar.

```
<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>
```

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

Visite o repositório central do Maven para obter o valor mais recente da [VERSAO](#).

 Note

Se vários clientes de serviço forem declarados em um pom.xml arquivo, todos exigirão o elemento exclusions XML.

Definido por meio da propriedade do sistema Java

Para usar o cliente HTTP AWS baseado em CRT como o HTTP padrão para seu aplicativo, você pode definir a propriedade do sistema Java software.amazon.awssdk.http.async.service.impl com um valor de software.amazon.awssdk.http.crt.AwsCrtSdkHttpService

Para definir durante a inicialização do aplicativo, execute um comando semelhante ao seguinte.

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

Use o seguinte trecho de código para definir a propriedade do sistema no código do seu aplicativo.

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

 Note

Você precisa adicionar uma dependência do aws-crt-client artefato em seu pom1.xml arquivo ao usar uma propriedade do sistema para configurar o uso do cliente HTTP baseado em AWS CRT.

Configuração avançada do **AwsCrtAsyncHttpClient**

Você pode usar o cliente HTTP AWS baseado em CRT para definir várias configurações, incluindo a configuração de integridade da conexão e o tempo máximo de inatividade. Você pode revisar as [opções de configuração disponíveis](#) para AwsCrtAsyncHttpClient o.

Configuração de integridade da conexão

Você pode definir a configuração da integridade da conexão para o cliente HTTP AWS baseado em CRT usando o `connectionHealthConfiguration` método no construtor de cliente HTTP.

O exemplo a seguir cria uma `S3AsyncClient` que usa uma `AwsCrtAsyncHttpClient` instância configurada com a configuração de integridade da conexão e um tempo máximo de inatividade para conexões.

```
// 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();
```

Suporte a HTTP/2

O protocolo HTTP/2 ainda não é suportado no cliente HTTP AWS baseado em CRT, mas está planejado para uma versão futura.

Enquanto isso, se você estiver usando clientes de serviço que exigem suporte HTTP/2, como o [KinesisAsyncClient](#) ou o [TranscribeStreamingAsyncClient](#), considere usar o [NettyNioAsyncHttpClient](#).

Exemplo de configuração de proxy

O trecho de código a seguir usa o [construtor de configuração de proxy para o cliente HTTP baseado em AWS CRT](#).

```
SdkAsyncHttpClient crtAsyncHttpClient = AwsCrtAsyncHttpClient.builder()  
    .proxyConfiguration(ProxyConfiguration.builder()  
        .scheme("https")  
        .host("myproxy")
```

```
.port(1234)
.username("username")
.password("password")
.build()
.build();
```

As propriedades equivalentes do sistema Java para a configuração do proxy são mostradas no seguinte trecho de linha de comando.

```
$ java -Dhttps.proxyHost=myproxy -Dhttps.proxyPort=1234 -Dhttps.proxyUser=username \
-Dhttps.proxyPassword=password -cp ... App
```

Important

Para usar qualquer uma das propriedades do sistema proxy HTTPS, a `scheme` propriedade deve ser definida em código para `https`. Se a propriedade do esquema não estiver definida no código, o esquema assumirá como padrão HTTP e o SDK `http.*` procurará somente as propriedades do sistema.

Tratamento de exceções para o AWS SDK for Java 2.x

Compreender como e quando o AWS SDK for Java 2.x lança exceções é importante para compilar aplicativos de alta qualidade usando o SDK. As seções a seguir descrevem os casos diferentes de exceções lançadas pelo SDK e como processá-las da maneira apropriada.

Por que exceções não verificadas?

O AWS SDK for Java usa exceções de tempo de execução (ou desmarcadas), em vez de exceções marcadas por estes motivos:

- Como permitir que desenvolvedores controlem os erros que desejam processar sem forçá-los a processar casos excepcionais com os quais não estão preocupados (e tornar o código excessivamente detalhado)
- Para evitar problemas de escalabilidade inerentes a exceções marcadas em aplicativos grandes

Em geral, as exceções marcadas funcionam bem em escalas pequenas, mas podem se tornar problemáticas à medida que os aplicativos crescem e se tornam mais complexos.

AwsServiceException (e subclasses)

[AwsServiceException](#) é a exceção mais comum que você enfrentará ao usar AWS SDK for Java o. [AwsServiceException](#) é uma subclasse das mais gerais [SdkServiceException](#). [AwsServiceExceptions](#) representam uma resposta de erro de um AWS service (Serviço da AWS). Por exemplo, se você tentar encerrar uma Amazon EC2 instância que não existe, Amazon EC2 retornará uma resposta de erro e todos os detalhes dessa resposta de erro serão incluídos na [AwsServiceException](#) resposta lançada.

Ao encontrar um [AwsServiceException](#), você sabe que sua solicitação foi enviada com sucesso para o AWS service (Serviço da AWS), mas não pôde ser processada com sucesso. Isso pode ocorrer devido a erros nos parâmetros da solicitação ou problemas no lado do serviço.

[AwsServiceException](#) fornece informações como:

- Código de status HTTP retornado
- Código de AWS erro retornado
- Mensagem de erro detalhada do serviço na [AwsErrorDetails](#) classe
- AWSID de solicitação para a solicitação que falhou

Na maioria dos casos, uma subclasse específica do serviço de [AwsServiceException](#) é lançada para permitir que os desenvolvedores tenham um controle refinado sobre como lidar com casos de erro por meio de catch blocks. A referência da API Java SDK para [AwsServiceException](#) exibe o grande número de [AwsServiceException](#) subclasses. Use os links da subclasse para detalhar e ver as exceções granulares lançadas por um serviço.

Por exemplo, os links a seguir para a referência da API do SDK mostram as hierarquias de exceções de alguns exemplos comuns. Serviços da AWS A lista de subclasses mostrada em cada página mostra as exceções específicas que seu código pode capturar.

- [Amazon S3](#)
- [DynamoDB](#)
- [Amazon SQS](#)

Para saber mais sobre uma exceção, inspecione `errorCode` o [AwsErrorDetails](#) objeto. Você pode usar o `errorCode` valor para pesquisar informações na API do guia de serviços. Por

exemplo, se um S3Exception for detectado e o AwsErrorDetails#errorCode() valor forInvalidRequest, use a [lista de códigos de erro](#) na Referência da API do Amazon S3 para ver mais detalhes.

SdkClientException

[SdkClientException](#) indica que ocorreu um problema dentro do código do cliente Java, ao tentar enviar uma solicitação para AWS ou ao tentar analisar uma resposta de AWS. Um geralmente SdkClientException é mais grave do que um SdkServiceException e indica um grande problema que está impedindo o cliente de fazer chamadas de serviço para AWS os serviços. Por exemplo, o AWS SDK for Java lançará um SdkClientException se nenhuma conexão de rede estiver disponível quando você tentar chamar uma operação em um dos clientes.

Exceções e comportamento de repetição

O SDK for Java repete solicitações para [várias exceções do lado do cliente](#) e [para códigos de status HTTP](#) que ele recebe das respostas. AWS service (Serviço da AWS) Esses erros são tratados como parte do legado RetryMode que os clientes de serviço usam por padrão. A referência da API Java para [RetryMode](#) descreve as várias maneiras pelas quais você pode configurar o modo.

Para personalizar as exceções e os códigos de status HTTP que acionam novas tentativas automáticas, configure seu cliente de serviço com um [RetryPolicy](#) que adicione instâncias [RetryOnExceptionsCondition](#), [RetryOnStatusCodeCondition](#)

Registro com o SDK for Java 2.x

AWS SDK for Java 2.x Ele usa [SLF4J](#), que é uma camada de abstração que permite o uso de qualquer um dos vários sistemas de registro em tempo de execução.

Os sistemas de registro suportados incluem o Java Logging Framework e o Apache [Log4j 2](#), entre outros. Este tópico mostra como usar o Log4j 2 como sistema de registro para trabalhar com o SDK.

Arquivo de configuração do Log4j 2

Normalmente, você usa um arquivo de configuração, nomeado log4j2.xml com Log4j 2. Os arquivos de configuração de exemplo são mostrados abaixo. Para saber mais sobre os valores usados no arquivo de configuração, consulte o [manual de configuração do Log4j](#).

O log4j2.xml arquivo precisa estar no classpath quando seu aplicativo é inicializado. Para um projeto Maven, coloque o arquivo no <project-dir>/src/main/resources diretório.

O arquivo de log4j2.xml configuração especifica propriedades como o [nível de registro](#), para onde a saída de registro é enviada (por exemplo, [para um arquivo ou para o console](#)) e o [formato da saída](#). O nível de registro especifica o nível de detalhe que o Log4j 2 gera. [O Log4j 2 suporta o conceito de várias hierarquias de registro](#). O nível de registro em log é definido de maneira independente para cada hierarquia. A hierarquia de registro principal que você usa com o AWS SDK for Java 2.x é software.amazon.awssdk.

Adicionar dependência de registro

Para configurar a associação Log4j 2 para SLF4J em seu arquivo de compilação, use o seguinte.

Maven

Adicione os seguintes elementos ao seu pom.xml arquivo.

```
...
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j2-impl</artifactId>
    <version>VERSION</version>
</dependency>
...
```

Gradle–Kotlin DSL

Adicione o seguinte ao seu build.gradle.kts arquivo.

```
...
dependencies {
    ...
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl:VERSION")
    ...
}
```

Use 2.20.0 para a versão mínima do log4j-slf4j2-impl artefato. Para a versão mais recente, use a versão publicada no [Maven central](#). Substitua **VERSION** pela versão que você usará.

Erros e avisos específicos do SDK

Recomendamos que você sempre deixe a hierarquia do registrador “software.amazon.awssdk” definida como “WARN” para capturar qualquer mensagem importante das bibliotecas de cliente do SDK. Por exemplo, se o cliente Amazon S3 detectar que seu aplicativo não fechou corretamente InputStream e pode estar vazando recursos, o cliente S3 reportará isso por meio de uma mensagem de aviso aos registros. Isso também garante que as mensagens serão registradas em log se o cliente enfrentar algum problema ao processar requisições ou respostas.

O log4j2.xml arquivo a seguir define o rootLogger como “WARN”, o que faz com que mensagens de aviso e erro de todos os registradores do aplicativo sejam enviadas, incluindo aqueles na hierarquia “software.amazon.awssdk”. Como alternativa, você pode definir explicitamente a hierarquia do registrador “software.amazon.awssdk” como “WARN” se for usada. <Root level="ERROR">

Exemplo de arquivo de configuração Log4j2.xml

Essa configuração registrará mensagens nos níveis “ERROR” e “WARN” no console para todas as hierarquias de registradores.

```
<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>
```

Registro resumido da solicitação/resposta

Cada solicitação para um AWS service (Serviço da AWS) gera um ID de AWS solicitação exclusivo que é útil se você tiver problemas com a forma como um AWS service (Serviço da AWS) está lidando com uma solicitação. AWSOs IDs de solicitação podem ser acessados programaticamente por meio de [SdkServiceException](#) objetos no SDK para qualquer chamada de serviço com

falha e também podem ser relatados por meio do nível de registro “DEBUG” do registrador “software.amazon.awssdk.request”.

O log4j2.xml arquivo a seguir permite um resumo das solicitações e respostas.

```
<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>
```

Aqui está um exemplo da saída do log:

```
2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
  DefaultSdkHttpRequest(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:
  QS9DUMME2NHEDH8TGT9N5V530JVV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not
  available
```

Se você estiver interessado apenas no ID da solicitação, use<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />.

Registro detalhado de cabos

Pode ser útil ver as solicitações e respostas exatas que o SDK for Java 2.x envia e recebe. Se precisar acessar essas informações, você pode ativá-las temporariamente adicionando a configuração necessária, dependendo do cliente HTTP usado pelo cliente de serviço.

Por padrão, clientes de serviços síncronos, como o [S3Client](#), usam um Apache subjacente, e clientes de serviços assíncronos [HttpClient](#), como o [S3 AsyncClient](#), usam um cliente HTTP Netty sem bloqueio.

Aqui está um resumo dos clientes HTTP que você pode usar para as duas categorias de clientes de serviço:

Clientes HTTP síncronos	Clientes HTTP assíncronos
ApacheHttpClient (padrão)	NettyNioAsyncHttpClient (padrão)
URLConnectionHttpClient	AwsCrtAsyncHttpClient

Consulte a guia apropriada abaixo para ver as configurações que você precisa adicionar, dependendo do cliente HTTP subjacente.

Warning

Recomendamos que você use o arquivo de log somente para fins de depuração. Desative-o em seus ambientes de produção, pois ele pode registrar dados confidenciais. Ele registra a solicitação ou resposta completa sem criptografia, até mesmo para uma chamada HTTPS. Para grandes solicitações (por exemplo, para fazer upload de um arquivo para o Amazon S3) ou respostas, o log detalhado da conexão também pode afetar significativamente o desempenho do aplicativo.

ApacheHttpClient

Adicione o registrador “org.apache.http.wire” ao arquivo de `log4j2.xml` configuração e defina o nível como “DEBUG”.

O `log4j2.xml` arquivo a seguir ativa o registro completo de conexões para o 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>
```

Uma dependência adicional do Maven no log4j-1.2-api artefato é necessária para o registro de conexões com o Apache, pois ele usa 1.2 sob o capô.

O conjunto completo de dependências do Maven para o log4j 2, incluindo o registro de conexões para o cliente Apache HTTP, é mostrado nos trechos do arquivo de compilação a seguir.

Maven

```
...
<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 is needed for Log4j2 with SLF4J -->
<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>
...
```

DSL Gradle—Kotlin

```
...
dependencies {
    ...
    implementation(platform("org.apache.logging.log4j:log4j-bom:VERSION"))
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")
    implementation("org.apache.logging.log4j:log4j-1.2-api")
}
...
```

Use 2.20.0 para a versão mínima do log4j-bom artefato. Para a versão mais recente, use a versão publicada no [Maven central](#). Substitua **VERSION** pela versão que você usará.

URLConnectionHttpClient

Para registrar detalhes de clientes de serviço que usam o `URLConnectionHttpClient`, primeiro crie um `logging.properties` arquivo com o seguinte conteúdo:

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

Defina a seguinte propriedade do sistema JVM com o caminho completo do:
`logging.properties`

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

Essa configuração registrará somente os cabeçalhos da solicitação e da resposta, por exemplo:

```
<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: sAFeZDOKdUMsBbkdyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFu0vr2tUb7Y1wEHGdJ3DSIxq0=}{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}
```

Para ver os corpos da solicitação/resposta, adicione `-Djavax.net.debug=all` às propriedades da JVM. Essa propriedade adicional registra uma grande quantidade de informações, incluindo todas as informações de SSL.

No console de log ou no arquivo de log, pesquise "GET" ou acesse rapidamente "POST" a seção do log que contém solicitações e respostas reais. "Plaintext before ENCRYPTION" Pesquise solicitações e respostas "Plaintext after DECRYPTION" para ver o texto completo dos cabeçalhos e corpos.

NettyNioAsyncHttpClient

Se seu cliente de serviço assíncrono usa o padrão `NettyNioAsyncHttpClient`, adicione dois registradores adicionais ao seu `log4j2.xml` arquivo para registrar cabeçalhos HTTP e corpos de solicitação/resposta.

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

Aqui está um `log4j2.xml` exemplo completo:

```
<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>

```

Essas configurações registram todos os detalhes do cabeçalho e corpos de solicitação/resposta.

AwsCrtAsyncHttpClient

Se você configurou seu cliente de serviço para usar uma instância `doAwsCrtAsyncHttpClient`, você pode registrar detalhes definindo as propriedades do sistema JVM ou programaticamente.

Log to a file at "Debug" level

Usando as propriedades do sistema:

```

-Daws.crt.log.level=Trace
-Daws.crt.log.destination=File
-Daws.crt.log.filename=<path to file>

```

Programaticamente:

```

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

Usando as propriedades do sistema:

```

-Daws.crt.log.level=Trace
-Daws.crt.log.destination=Stdout

```

Programaticamente:

```

import software.amazon.awssdk.crt.Log;

// Execute this statement before constructing the
// SDK service client.
Log.initLoggingToStdout(Log.LogLevel.Trace);

```

Por motivos de segurança, no nível “Rastrear”, os `AwsCrtAsyncHttpClient` registros somente são os cabeçalhos de resposta. Cabeçalhos de solicitação, corpos de solicitação e corpos de resposta não são registrados.

Definir o TTL da JVM para pesquisas de nomes DNS

A JVM armazena em cache pesquisas de nome DNS. Quando a JVM resolve um nome de host para um endereço IP, ela armazena o endereço IP em cache por um período de tempo especificado, conhecido como time-to-live(TTL).

Como os recursos AWS usam entradas de nome DNS que acabam mudando, recomendamos configurar a JVM com um valor TTL de até 60 segundos. Isso garante que, quando o endereço IP de um recurso mudar, o aplicativo poderá receber e usar o novo endereço IP do recurso consultando novamente o DNS.

Em algumas configurações do Java, o TTL padrão da JVM é definido de maneira que jamais atualizará entradas DNS até a JVM ser reiniciada. Assim, se o endereço IP de um AWS resource muda enquanto seu aplicativo ainda está em execução, ele não poderá usar esse recurso até que você reiniciar manualmente a JVM e as informações de IP em cache são atualizadas. Nesse caso, é crucial definir o TTL da JVM, de forma que ele atualize periodicamente as informações de IP armazenadas em cache.

Note

O TTL padrão pode variar de acordo com a versão da JVM e a possibilidade de um [gerenciador de segurança](#) estar instalado. Muitas JVMs oferecem um TTL padrão menor que 60 segundos. Se estiver usando uma JVM como essa, e não um gerenciador de segurança, será possível ignorar o restante deste tópico.

Como configurar o TTL da JVM

Para modificar o TTL da JVM, defina o valor da propriedade [`networkaddress.cache.ttl`](#). Use um dos seguintes métodos, dependendo das necessidades:

- globalmente, para todos os aplicativos que usam a JVM. Defina `networkaddress.cache.ttl` no arquivo `$JAVA_HOME/jre/lib/security/java.security`:

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

- somente para o aplicativo, defina `networkaddress.cache.ttl` no código de inicialização do aplicativo:

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

Melhores recomendas para o AWS SDK for Java 2.x

Esta seção lista as melhores práticas de uso do SDK for Java 2.x.

Tópicos

- [Reutilize um cliente SDK, se possível](#)
- [Feche fluxos de entrada das operações do cliente](#)
- [Ajuste as configurações HTTP com base em testes de desempenho](#)
- [Use o OpenSSL para o cliente HTTP baseado em Netty](#)
- [Configurar tempos limite de API](#)
- [Usar métricas](#)

Reutilize um cliente SDK, se possível

Cada cliente SDK mantém seu próprio pool de conexões HTTP. Uma conexão que já existe no pool pode ser reutilizada por uma nova solicitação para reduzir o tempo de estabelecimento de uma nova conexão. Recomendamos compartilhar uma única instância do cliente para evitar a sobrecarga de ter muitos pools de conexões que não são usados de forma eficaz. Todos os clientes do SDK estão seguros.

Se você não quiser compartilhar uma instância do cliente, chame `close()` a instância para liberar os recursos quando o cliente não for necessário.

Feche fluxos de entrada das operações do cliente

Para operações de streaming [`S3Client#getObject`](#), como, se você estiver trabalhando [`ResponseInputStream`](#) diretamente com, recomendamos fazer o seguinte:

- Leia todos os dados do fluxo de entrada o mais rápido possível.

- Feche o fluxo de entrada o mais rápido possível.

Fazemos essas recomendações porque o fluxo de entrada é um fluxo direto de dados da conexão HTTP e a conexão HTTP subjacente não pode ser reutilizada até que todos os dados do fluxo tenham sido lidos e o fluxo seja fechado. Se essas regras não forem seguidas, o cliente poderá ficar sem recursos alocando muitas conexões HTTP abertas, mas não utilizadas.

Ajuste as configurações HTTP com base em testes de desempenho

O SDK fornece um conjunto de [configurações http padrão](#) que se aplicam a casos de uso gerais. Recomendamos que os clientes ajustem as configurações HTTP de seus aplicativos com base em seus casos de uso.

Como um bom ponto de partida, o SDK oferece um recurso [inteligente de padrões de configuração](#). Esse recurso está disponível a partir da versão 2.17.102. Escolha um modo de acordo com seu caso de uso, que fornece valores de configuração sensatos.

Use o OpenSSL para o cliente HTTP baseado em Netty

Por padrão, o SDK [NettyNioAsyncHttpClient](#) usa a implementação SSL padrão do JDK comoSslProvider o. Nossos testes descobriram que o OpenSSL tem um desempenho melhor do que a implementação padrão do JDK. A comunidade Netty também [recomenda o uso do OpenSSL](#).

Para usar o OpenSSL, adicione netty-tcnative às suas dependências. Para obter detalhes de configuração, consulte a [documentação do projeto Netty](#).

Depois de netty-tcnative configurar para seu projeto, a NettyNioAsyncHttpClient instância selecionará automaticamente o OpenSSL. Como alternativa, você pode definir oSslProvider explicitamente usando o NettyNioAsyncHttpClient construtor, conforme mostrado no trecho a seguir.

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

Configurar tempos limite de API

O SDK fornece [valores padrão](#) para algumas opções de tempo limite, como tempo limite de conexão e tempo limite de soquete, mas não para tempos limite de chamadas de API ou tempos limite de

tentativas de chamada de API individuais. É uma boa prática definir tempos limite para as tentativas individuais e para toda a solicitação. Isso garantirá que seu aplicativo falhe rapidamente de maneira ideal quando houver problemas transitórios que possam fazer com que as tentativas de solicitação levem mais tempo para serem concluídas ou problemas de rede fatais.

Você pode configurar tempos limite para todas as solicitações feitas por um cliente de serviço usando [ClientOverrideConfiguration#apiCallAttemptTimeout](#) e [ClientOverrideConfiguration#apiCallTimeout](#).

O exemplo a seguir mostra a configuração de um cliente Amazon S3 com valores de tempo limite personalizados.

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

apiCallAttemptTimeout

Essa configuração define a quantidade de tempo para uma única tentativa de HTTP, após a qual a chamada de API pode ser repetida.

apiCallTimeout

O valor dessa propriedade configura a quantidade de tempo de toda a execução, incluindo todas as tentativas de nova tentativa.

Como alternativa à definição desses valores de tempo limite no cliente de serviço, você pode [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) usar [RequestOverrideConfiguration#apiCallTimeout\(\)](#) e configurar uma única solicitação.

O exemplo a seguir configura uma única `listBuckets` solicitação com valores de tempo limite personalizados.

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

Ao usar essas propriedades juntas, você define um limite rígido para o tempo total gasto em todas as tentativas em novas tentativas. Você também define uma solicitação HTTP individual para falhar rapidamente em uma solicitação lenta.

Usar métricas

O SDK for Java pode [coletar métricas](#) para os clientes de serviço em seu aplicativo. Você pode usar essas métricas para identificar problemas de desempenho, analisar tendências gerais de uso, analisar as exceções retornadas pelo cliente de serviço ou para entender um problema específico.

Recomendamos que você colete métricas e, em seguida, analise os Amazon CloudWatch Logs para obter uma compreensão mais profunda do desempenho do seu aplicativo.

Características do AWS SDK for Java 2.x

Esta seção fornece informações sobre os recursos do AWS SDK for Java 2.x.

Tópicos

- [Programação assíncrona](#)
- [API de cliente aprimorada do DynamoDB no AWS SDK for Java 2.x](#)
- [Trabalhando com HTTP/2 no AWS SDK for Java](#)
- [Ative as métricas do SDK para o AWS SDK for Java](#)
- [Recuperando resultados paginados usando o 2.x AWS SDK for Java](#)
- [AWSClient S3 baseado em CRT](#)
- [Gerenciador de transferências do Amazon S3](#)
- [Use garçons no 2.x AWS SDK for Java](#)
- [API IAM Policy Builder](#)

Programação assíncrona

O AWS SDK for Java 2.x apresenta clientes assíncronos verdadeiramente sem bloqueio que implementam alta simultaneidade em alguns segmentos. O AWS SDK for Java 1.x tem clientes assíncronos que envolvem um pool de threads e bloqueiam clientes síncronos que não oferecem todos os benefícios da E/S sem bloqueio.

Os métodos síncronos bloqueiam a execução do seu thread até o cliente receber uma resposta do serviço. Os métodos assíncronos retornam imediatamente, devolvendo o controle ao thread de chamada sem aguardar uma resposta.

Como um método assíncrono retorna antes de uma resposta estar disponível, você precisa de uma maneira de obter a resposta quando ela estiver pronta. Os métodos para clientes assíncronos em 2.x dos `CompletableFuture` objetos de AWS SDK for Java retorno que permitem acessar a resposta quando ela estiver pronta.

Operações sem streaming

Para as operações que não são de streaming, as chamadas de método assíncrono são semelhantes aos métodos síncronos. No entanto, os métodos assíncronos no AWS SDK for Java retornam um `CompletableFuture` objeto que contém os resultados da operação assíncrona no futuro.

Chame o `CompletableFuture whenComplete()` método com uma ação a ser concluída quando o resultado estiver disponível. `CompletableFuture` implementa a `Future` interface, para que você também possa obter o objeto de resposta chamando o `get()` método.

Veja a seguir um exemplo de uma operação assíncrona que chama uma Amazon DynamoDB função para obter uma lista de tabelas, recebendo uma `CompletableFuture` que pode conter um objeto. [ListTablesResponse](#) A ação definida na chamada para `whenComplete()` é feita somente quando a chamada assíncrona é concluída.

Importações

```
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;
```

Código

```
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();
}
}
```

O exemplo de código a seguir mostra como recuperar um item de uma tabela usando o cliente assíncrono. Invoca o `getItem` método do `DynamoDbAsyncClient` e passe a ele um [GetItemRequest](#) objeto com o nome da tabela e o valor da chave primária do item desejado. Normalmente, é assim que você passa dados exigidos pela operação. Neste exemplo, observe que um valor String é passado.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Operações de streaming

Para operações de streaming, você deve fornecer um [AsyncRequestBody](#) para fornecer o conteúdo de forma incremental ou um [AsyncResponseTransformer](#) para receber e processar a resposta.

O exemplo a seguir faz upload de um arquivo para o Amazon S3 de forma assíncrona usando a operação PutObject.

Importações

```
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;
```

Código

```
/** 
 * 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();
}
}
```

O exemplo a seguir obtém um arquivo do Amazon S3 de forma assíncrona usando a operação `GetObject`.

Importações

```
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;
```

Código

```
/***
 * 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();
}
}
```

Operações avançadas

O AWS SDK for Java 2.x usa o [Netty](#), uma estrutura assíncrona de aplicativos de rede orientada por eventos, para lidar com threads de E/S. O AWS SDK for Java 2.x cria um Netty ExecutorService por trás, para completar os futuros retornados da solicitação do cliente HTTP para o cliente Netty. Essa abstração reduz o risco de um aplicativo quebrar o processo assíncrono se os desenvolvedores optarem por interromper ou suspender threads. Por padrão, cada cliente assíncrono cria um threadpool com base no número de processadores e gerencia as tarefas em uma fila dentro do ExecutorService

Os usuários avançados podem especificar o tamanho do grupo de threads ao criar um cliente assíncrono usando a opção a seguir durante a compilação.

Código

```
S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
```

```
.FUTURE_COMPLETION_EXECUTOR,  
Executors.newFixedThreadPool(10)  
)  
)  
.build();
```

Para otimizar o desempenho, você pode gerenciar seu próprio executor de grupo de threads e incluí-lo ao configurar seu cliente.

```
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();
```

Se preferir não usar um grupo de threads, use `Runnable::run` em vez de usar um executor de grupo de threads.

```
S3AsyncClient clientThread = S3AsyncClient.builder()  
.asyncConfiguration(  
b -> b.advancedOption(SdkAdvancedAsyncClientOption  
.FUTURE_COMPLETION_EXECUTOR,  
Runnable::run  
)  
)  
.build();
```

API de cliente aprimorada do DynamoDB no AWS SDK for Java 2.x

A [DynamoDB Enhanced Client](#) API é uma biblioteca de alto nível que é a sucessora da classe de no SDK for DynamoDBMapper Java v1.x. Ele oferece uma maneira simples de mapear classes do lado do cliente para tabelas do DynamoDB. Defina as relações entre as tabelas e suas classes de modelo correspondentes no seu código. Depois que você definir essas relações, poderá executar intuitivamente várias operações de criação, leitura, atualização ou exclusão (CRUD) em tabelas ou itens do DynamoDB.

A API do DynamoDB Enhanced Client também inclui [a API de documentos aprimorada](#), que permite trabalhar com itens do tipo documento que não seguem um esquema definido.

A API do DynamoDB Enhanced Client é discutida nos tópicos a seguir.

- [Comece a usar a API do DynamoDB Enhanced Client](#)
- [Noções básicas da API de cliente aprimorada do DynamoDB](#)
- [Extensões](#)
- [Recursos avançados de esquema de tabela](#)
- [API de documentos aprimorada para DynamoDB](#)
- [Operações assíncronas sem bloqueio](#)
- [Anotações de classes de dados](#)

Comece a usar a API do DynamoDB Enhanced Client

O tutorial a seguir apresenta os fundamentos necessários para trabalhar com a API do DynamoDB Enhanced Client.

Adicionar dependências

Para começar a trabalhar com a API do DynamoDB Enhanced Client em seu projeto, adicione uma dependência no artefato Maven. dynamodb-enhanced Isso é mostrado nos exemplos a seguir.

Maven

```
<project>
  <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>dynamodb-enhanced</artifactId>
    </dependency>
</dependencies>
...
</project>
```

Faça uma pesquisa no repositório central do Maven para obter a [versão mais recente](#) e <VERSION> substitua por esse valor.

Gradle

```
repositories {
    mavenCentral()
}
dependencies {
    implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
    implementation("software.amazon.awssdk:dynamodb-enhanced")
    ...
}
```

Faça uma pesquisa no repositório central do Maven para obter a [versão mais recente](#) e <VERSION> substitua por esse valor.

Gerar um **TableSchema**

A [TableSchema](#) permite que o cliente aprimorado mapeie valores de atributos do DynamoDB de e para suas classes do lado do cliente. Neste tutorial, você aprende sobre TableSchema s derivado de uma classe de dados estática e gerado a partir de código usando um construtor.

Use uma classe de dados anotada

O SDK for Java 2.x inclui [um conjunto de](#) anotações que você pode usar com uma classe de dados para gerar rapidamente TableSchema uma para mapear suas classes em tabelas.

Comece criando uma classe de dados que esteja em conformidade com a [JavaBean especificação](#). A especificação exige que uma classe tenha um construtor público sem argumentos e tenha getters e setters para cada atributo na classe. Inclua uma anotação em nível de classe para indicar que a classe de dados é uma DynamoDbBean Além disso, no mínimo, inclua uma DynamoDbPartitionKey anotação no getter ou setter do atributo da chave primária.

Note

O termo `property` é normalmente usado para um valor encapsulado em um JavaBean. No entanto, este guia usa o termo `attribute` em vez disso, para ser consistente com a terminologia usada pelo DynamoDB.

A `Customer` classe a seguir mostra as anotações que vinculam a definição da classe à tabela do DynamoDB.

Classe `Customer`

```
package org.example.tests.model;

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;

@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 this.regDate; }

public void setRegistrationDate(Instant registrationDate) { this.regDate =
registrationDate; }

@Override
public String toString() {
    return "Customer [id=" + id + ", name=" + name + ", email=" + email
           + ", regDate=" + regDate + "]";
}

}
```

Depois de criar uma classe de dados anotada, use-a para criar aTableSchema, conforme mostrado no trecho a seguir.

```
static final TableSchema<Customer> customerTableSchema =
TableSchema.fromBean(Customer.class);
```

A foi TableSchema projetado para ser estático e imutável. Normalmente, você pode instanciá-lo no momento do carregamento da classe.

O método static TableSchema.fromBean() factory faz uma introspecção do bean para gerar o mapeamento dos atributos da classe de dados de e para os atributos do DynamoDB.

Para ver um exemplo de como trabalhar com um modelo de dados composto por várias classes de dados, consulte a Person classe na [???](#) seção.

Use um construtor

Você pode ignorar o custo da introspecção do bean se definir o esquema da tabela no código. Se você codificar o esquema, sua classe não precisará seguir os padrões de JavaBean nomenclatura

nem precisará ser anotada. O exemplo a seguir usa um construtor e é equivalente ao exemplo de `Customer` classe que usa anotações.

```
static final TableSchema<Customer> customerTableSchema =  
    TableSchema.builder(Customer.class)  
        .newItemSupplier(Customer::new)  
        .addAttribute(String.class, a -> a.name("id")  
            .getter(Customer::getId)  
            .setter(Customer::setId)  
            .tags(StaticAttributeTags.primaryPartitionKey()))  
        .addAttribute(String.class, a -> a.name("email")  
            .getter(Customer::getEmail)  
            .setter(Customer::setEmail)  
            .tags(StaticAttributeTags.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();
```

Crie um cliente aprimorado e **DynamoDbTable**

Crie um cliente aprimorado

A [DynamoDbEnhancedClient](#) classe, ou sua contraparte assíncrona, [DynamoDbEnhancedAsyncClient](#), é o ponto de entrada para trabalhar com a API do DynamoDB Enhanced Client.

O cliente aprimorado exige um padrão [DynamoDbClient](#) para realizar o trabalho. A API oferece duas maneiras de criar uma `DynamoDbEnhancedClient` instância. A primeira opção, mostrada no trecho a seguir, cria um padrão `DynamoDbClient` com configurações padrão retiradas das configurações.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```

Se quiser configurar o cliente padrão subjacente, você pode fornecê-lo ao método construtor do cliente aprimorado, conforme mostrado no trecho a seguir.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
```

```
.dynamoDbClient(  
    // Configure an instance of the standard client.  
    DynamoDbClient.builder()  
        .region(Region.US_EAST_1)  
  
.credentialsProvider(ProfileCredentialsProvider.create())  
    .build()  
    .build();
```

Crie uma **DynamoDbTable** instância

Pense em a [DynamoDbTable](#) como a representação do lado do cliente de uma tabela do DynamoDB que usa a funcionalidade de mapeamento fornecida por a. TableSchema A DyanamoDbTable classe fornece métodos para operações CRUD que permitem que você interaja com uma única tabela do DynamoDB.

DynamoDbTable<T>é uma classe genérica que usa um argumento de tipo único, seja uma classe personalizada ou EnhancedDocument ao trabalhar com itens do tipo documento. Esse tipo de argumento estabelece a relação entre a classe que você usa e a tabela única do DynamoDB.

Use o método de `table()` fábrica do `DynamoDbEnhancedClient` para criar uma `DynamoDbTable` instância, conforme mostrado no trecho a seguir.

```
static final DynamoDbTable<Customer> customerTable =  
    enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

`DynamoDbTable`instâncias são candidatas a `singletons` porque são imutáveis e podem ser usadas em toda a sua aplicação.

Seu código agora tem uma representação na memória de uma tabela do DynamoDB que pode armazenar instâncias. `Customer` A tabela real do DynamoDB pode ou não existir. Se a tabela nomeada `Customer` já existir, você poderá começar a realizar operações CRUD nela. Se ela não existir, use a `DynamoDbTable` instância para criar a tabela conforme discutido na próxima seção.

Crie uma tabela do DynamoDB, se necessário

Depois de criar uma `DynamoDbTable` instância, use-a para realizar uma criação única de uma tabela no DynamoDB.

Crie um código de exemplo de tabela

O exemplo a seguir cria uma tabela do DynamoDB com base na `Customer` classe de dados.

Este exemplo cria uma tabela do DynamoDB com o `Customer` nome — idêntico ao nome da classe — mas o nome da tabela pode ser diferente. Seja qual for o nome da tabela, você deve usar esse nome em aplicativos adicionais para trabalhar com a tabela. Forneça esse nome ao `table()` método sempre que criar outro `DynamoDbTable` objeto para trabalhar com a tabela subjacente do DynamoDB.

O parâmetro Java `lambda.builder`, passado para o `createTable` método permite que você [personalize a tabela](#). Neste exemplo, a taxa de [transferência provisionada](#) está configurada. Se você quiser usar as configurações padrão ao criar uma tabela, ignore o construtor, conforme mostrado no trecho a seguir.

```
customerDynamoDbTable.createTable();
```

Quando as configurações padrão são usadas, os valores para a taxa de transferência provisionada não são definidos. Em vez disso, o modo de cobrança da tabela é definido como [sob demanda](#).

O exemplo também usa um [DynamoDbWaiter](#) antes de tentar imprimir o nome da tabela recebida na resposta. A criação de uma tabela leva algum tempo. Portanto, usar um garçom significa que você não precisa escrever uma lógica que pesquise o serviço do DynamoDB para ver se a tabela existe antes de usá-la.

Importações

```
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;
```

Código

```
public static void createCustomerTable(DynamoDbTable<Customer> customerDynamoDbTable,
DynamoDbClient dynamoDbClient) {
    // Create the DynamoDB table by using the 'customerDynamoDbTable' DynamoDbTable
    // instance.
    customerDynamoDbTable.createTable(builder -> builder
        .provisionedThroughput(b -> b
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
```

```
        .build())
);
// The 'dynamoDbClient' instance that's passed to the builder for the
DynamoDbWaiter is the same instance
// that was passed to the builder of the DynamoDbEnhancedClient instance used to
create the 'customerDynamoDbTable'.
// This means that the same Region that was configured on the standard
'dynamoDbClient' instance is used for all service clients.
try (DynamoDbWaiter waiter =
DynamoDbWaiter.builder().client(dynamoDbClient).build()) { // DynamoDbWaiter is
Autocloseable
    ResponseOrException<DescribeTableResponse> response = waiter
        .waitForTableExists(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()
    logger.info("Customer table was created.");
}
}
```

Note

Os nomes dos atributos de uma tabela do DynamoDB começam com uma letra minúscula quando a tabela é gerada a partir de uma classe de dados. Se você quiser que o nome do atributo da tabela comece com uma letra maiúscula, use a [@DynamoDbAttribute\(NAME\) anotação](#) e forneça o nome desejado como parâmetro.

Execute operações

Depois que a tabela for criada, use a `DynamoDbTable<Customer>` instância para realizar operações na tabela do DynamoDB.

No exemplo a seguir, um singleton `DynamoDbTable<Customer>` é passado como parâmetro junto com uma instância de [classe de Customer dados](#) para adicionar um novo item à tabela.

```
public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
    logger.info(customer.toString());
    customerTable.putItem(customer);
```

```
}
```

Objeto `Customer`

```
Customer customer = new Customer();
customer.setId("1");
customer.setCustName("Customer Name");
customer.setEmail("customer@example.com");
customer.setRegistrationDate(Instant.parse("2023-07-03T10:15:30.00Z"));
```

Antes de enviar o `customer` objeto para o serviço do DynamoDB, registre a saída do método `toString()` do objeto para compará-la com o que o cliente aprimorado envia.

```
Customer [id=1, name=Customer Name, email=customer@example.com,
regDate=2023-07-03T10:15:30Z]
```

O registro em nível de fio mostra a carga útil da solicitação gerada. O cliente aprimorado gerou a representação de baixo nível da classe de dados. O `regDate` atributo, que é um `Instant` tipo em Java, é representado como uma string do DynamoDB.

```
{
    "TableName": "Customer",
    "Item": {
        "registrationDate": {
            "S": "2023-07-03T10:15:30Z"
        },
        "id": {
            "S": "1"
        },
        "custName": {
            "S": "Customer Name"
        },
        "email": {
            "S": "customer@example.com"
        }
    }
}
```

Trabalhar com uma tabela existente

A seção anterior mostrou como criar uma tabela do DynamoDB começando com uma classe de dados Java. Se você já tiver uma tabela existente e quiser usar os recursos do cliente aprimorado, poderá criar uma classe de dados Java para trabalhar com a tabela. Você precisa examinar a tabela do DynamoDB e adicionar as anotações necessárias à classe de dados.

Antes de trabalhar com uma tabela existente, chame o `DynamoDbEnhanced.table()` método. Isso foi feito no exemplo anterior com a seguinte declaração.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",  
    TableSchema.fromBean(Customer.class));
```

Depois que a `DynamoDbTable` instância for retornada, você poderá começar a trabalhar imediatamente com a tabela subjacente. Você não precisa recriar a tabela chamando o `DynamoDbTable.createTable()` método.

O exemplo a seguir demonstra isso recuperando imediatamente uma `Customer` instância da tabela do DynamoDB.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",  
    TableSchema.fromBean(Customer.class));  
// The Customer table exists already and has an item with a primary key value of "1"  
// and a sort key value of "customer@example.com".  
customerTable.getItem(  
    Key.builder()  
        .partitionValue("1").  
        sortValue("customer@example.com").build());
```

Important

O nome da tabela usado no `table()` método deve corresponder ao nome da tabela existente do DynamoDB.

Noções básicas da API de cliente aprimorada do DynamoDB

[Este tópico discute os recursos básicos da DynamoDB Enhanced Client API e a compara com a API de cliente padrão do DynamoDB.](#)

Se você não conhece a API do DynamoDB Enhanced Client, recomendamos que você leia o tutorial [introdutório para se familiarizar com as classes fundamentais](#).

Itens do DynamoDB em Java

As tabelas do DynamoDB armazenam itens. Dependendo do seu caso de uso, os itens no lado Java podem assumir a forma de dados estruturados estaticamente ou estruturas criadas dinamicamente.

Se seu caso de uso exigir itens com um conjunto consistente de atributos, use [classes anotadas](#) ou use um [construtor](#) para gerar a tipagem estática apropriada. TableSchema

Como alternativa, se você precisar armazenar itens que consistem em estruturas variadas, crie um DocumentTableSchema. DocumentTableSchema faz parte da [API de documentos aprimorados](#) e requer somente uma chave primária digitada estaticamente e funciona com EnhancedDocument instâncias para armazenar os elementos de dados. A API de documentos aprimorada é abordada em outro [tópico](#).

Tipos de atributos

Embora o DynamoDB [ofereça suporte a um pequeno número de tipos de atributos](#) em comparação com o sistema de tipos avançados do Java, a API do DynamoDB Enhanced Client fornece mecanismos para converter membros de uma classe Java de e para tipos de atributos do DynamoDB.

[Por padrão, a API do DynamoDB Enhanced Client suporta conversores de atributos para um grande número de tipos, como Integer, String e Instant. BigDecimal](#) A lista aparece nas [classes de implementação conhecidas da AttributeConverter interface](#). A lista inclui muitos tipos e coleções, como mapas, listas e conjuntos.

Para armazenar os dados de um tipo de atributo que não é suportado por padrão ou não está em conformidade com a JavaBean convenção, você pode escrever uma AttributeConverter implementação personalizada para fazer a conversão. Consulte a seção de conversão de atributos para ver um [exemplo](#).

Para armazenar os dados de um tipo de atributo cuja classe está em conformidade com a especificação Java Beans (ou uma [classe de dados imutável](#)), você pode adotar duas abordagens.

- Se você tiver acesso ao arquivo de origem, poderá anotar a classe com @DynamoDbBean (ou @DynamoDbImmutable). A seção que discute atributos aninhados mostra [exemplos](#) do uso de classes anotadas.

- Se você não tiver acesso ao arquivo de origem da classe de JavaBean dados do atributo (ou não quiser anotar o arquivo de origem de uma classe à qual você tem acesso), você pode usar a abordagem do construtor. Isso cria um esquema de tabela sem definir as chaves. Em seguida, você pode aninhar esse esquema de tabela dentro de outro esquema de tabela para realizar o mapeamento. A seção de atributos aninhados tem um [exemplo](#) que mostra o uso de esquemas aninhados.

Valores do tipo primitivo Java

Embora o cliente aprimorado possa trabalhar com atributos de tipos primitivos, incentivamos o uso de tipos de objetos porque você não pode representar valores nulos com tipos primitivos.

Valores nulos

Quando você usa a `putItem` API, o cliente aprimorado não inclui atributos de valor nulo de um objeto de dados mapeado na solicitação ao DynamoDB.

Para `updateItem` solicitações, atributos com valor nulo são removidos do item no banco de dados. Se você pretende atualizar alguns valores de atributos e manter os outros inalterados, copie os valores de outros atributos que não devem ser alterados ou use o método [ignoreNull\(\)](#) no construtor de atualizações.

O exemplo a seguir demonstra o uso do `updateItem()` método `ignoreNulls()`.

```
public void updateItemNullsExample(){
    Customer customer = new Customer();
    customer.setCustName("CustName");
    customer.setEmail("email");
    customer.setId("1");
    customer.setRegistrationDate(Instant.now());

    // Put item with values for all attributes.
    customerDynamoDbTable.putItem(customer);

    // Create a Customer instance with the same id value, but a different name
    // value.
    // Do not set the 'registrationDate' attribute.
    Customer custForUpdate = new Customer();
    custForUpdate.setCustName("NewName");
    custForUpdate.setEmail("email");
    custForUpdate.setId("1");
```

```
// Update item without setting the registrationDate attribute.  
customerDynamoDbTable.updateItem(b -> b  
    .item(custForUpdate)  
    .ignoreNulls(Boolean.TRUE));  
  
Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);  
// registrationDate value is unchanged.  
logger.info(updatedWithNullsIgnored.toString());  
  
customerDynamoDbTable.updateItem(custForUpdate);  
Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);  
// registrationDate value is null because ignoreNulls() was not used.  
logger.info(updatedWithNulls.toString());  
}  
}  
  
// Logged lines.  
Customer [id=1, custName=NewName, email=email,  
registrationDate=2023-04-05T16:32:32.056Z]  
Customer [id=1, custName=NewName, email=email, registrationDate=null]
```

Métodos básicos do DynamoDB Enhanced Client

Os métodos básicos do cliente aprimorado mapeiam as operações de serviço do DynamoDB que deram nome a eles. Os exemplos a seguir mostram a variação mais simples de cada método. Você pode personalizar cada método passando um objeto de solicitação aprimorado. Os objetos de solicitação aprimorados oferecem a maioria dos recursos disponíveis no cliente padrão do DynamoDB. Eles estão totalmente documentados na Referência AWS SDK for Java 2.x da API.

O exemplo usa o [the section called “Classe Customer”](#) mostrado anteriormente.

```
// CreateTable  
customerTable.createTable();  
  
// GetItem  
Customer customer =  
    customerTable.getItem(Key.builder().partitionValue("a123").build());  
  
// UpdateItem  
Customer updatedCustomer = customerTable.updateItem(customer);  
  
// PutItem  
customerTable.putItem(customer);
```

```
// DeleteItem
Customer deletedCustomer =
    customerTable.deleteItem(Key.builder().partitionValue("a123").sortValue(456).build());

// Query
PageIterable<Customer> customers = customerTable.query(keyEqualTo(k ->
    k.partitionValue("a123")));

// Scan
PageIterable<Customer> customers = customerTable.scan();

// BatchGetItem
BatchGetResultPageIterable batchResults =
    enhancedClient.batchGetItem(r -> r.addReadBatch(ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .add.GetItem(key1)
        .add.GetItem(key2)
        .add.GetItem(key3)
        .build()));

// BatchWriteItem
batchResults = enhancedClient.batchWriteItem(r ->
    r.addWriteBatch(WriteBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        .add.PutItem(customer)
        .add.DeleteItem(key1)
        .add.DeleteItem(key1)
        .build()));

// TransactGetItems
transactResults = enhancedClient.transactGetItems(r -> r.add.GetItem(customerTable,
    key1)
    .add.GetItem(customerTable,
    key2));

// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
    i -> i.key(orderKey)

    .conditionExpression(conditionExpression))
    .add.UpdateItem(customerTable, customer)
    .add.DeleteItem(customerTable, key));
```

Compare o DynamoDB Enhanced Client com o cliente DynamoDB padrão

As duas APIs de cliente do DynamoDB — [padrão e aprimoradas](#) — permitem que você trabalhe com tabelas do DynamoDB para realizar operações CRUD (criar, ler, atualizar e excluir) em nível de dados. A diferença entre as APIs do cliente é como isso é feito. Usando o cliente padrão, você trabalha diretamente com atributos de dados de baixo nível. A API de cliente aprimorada usa classes Java familiares e mapeia a API de baixo nível nos bastidores.

Embora ambas as APIs do cliente ofereçam suporte a operações em nível de dados, o cliente padrão do DynamoDB também oferece suporte a operações em nível de recursos. As operações em nível de recurso gerenciam o banco de dados, como criar backups, listar tabelas e atualizar tabelas. A API de cliente aprimorada oferece suporte a um número selecionado de operações em nível de recurso, como criar, descrever e excluir tabelas.

Para ilustrar as diferentes abordagens usadas pelas duas APIs do cliente, os exemplos de código a seguir mostram a criação da mesma `ProductCatalog` tabela usando o cliente padrão e o cliente aprimorado.

Compare: crie uma tabela usando o cliente padrão do DynamoDB

```
DependencyFactory.dynamoDbClient().createTable(builder -> builder
    .tableName(TABLE_NAME)
    .attributeDefinitions(
        b -> b.attributeName("id").attributeType(ScalarAttributeType.N),
        b -> b.attributeName("title").attributeType(ScalarAttributeType.S),
        b -> b.attributeName("isbn").attributeType(ScalarAttributeType.S)
    )
    .keySchema(
        builder1 -> builder1.attributeName("id").keyType(KeyType.HASH),
        builder2 -> builder2.attributeName("title").keyType(KeyType.RANGE)
    )
    .globalSecondaryIndexes(builder3 -> builder3
        .indexName("products_by_isbn")
        .keySchema(builder2 -> builder2
            .attributeName("isbn").keyType(KeyType.HASH))
        .projection(builder2 -> builder2
            .projectionType(ProjectionType.INCLUDE)
            .nonKeyAttributes("price", "authors"))
        .provisionedThroughput(builder4 -> builder4
            .writeCapacityUnits(5L).readCapacityUnits(5L))
    )
    .provisionedThroughput(builder1 -> builder1
```

```
        .readCapacityUnits(5L).writeCapacityUnits(5L))  
);
```

Compare: crie uma tabela usando o DynamoDB Enhanced Client

```
DynamoDbEnhancedClient enhancedClient = DependencyFactory.enhancedClient();  
productCatalog = enhancedClient.table(TABLE_NAME,  
    TableSchema.fromImmutableClass(ProductCatalog.class));  
productCatalog.createTable(b -> b  
    .provisionedThroughput(b1 -> b1.readCapacityUnits(5L).writeCapacityUnits(5L))  
    .globalSecondaryIndices(b2 -> b2.indexName("products_by_isbn"))  
    .projection(b4 -> b4  
        .projectionType(ProjectionType.INCLUDE)  
        .nonKeyAttributes("price", "authors"))  
    .provisionedThroughput(b3 ->  
        b3.writeCapacityUnits(5L).readCapacityUnits(5L))  
    )  
);
```

O cliente aprimorado usa a seguinte classe de dados anotada. O DynamoDB Enhanced Client mapeia tipos de dados Java para tipos de dados do DynamoDB para obter um código menos detalhado e mais fácil de seguir. ProductCatalog é um exemplo do uso de uma classe imutável com o DynamoDB Enhanced Client. O uso de classes imutáveis para classes de dados mapeados será [discutido posteriormente neste tópico](#).

Classe **ProductCatalog**

```
package org.example.tests.model;  
  
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbIgnore;  
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;  
import  
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;  
import  
    software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;  
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;  
  
import java.math.BigDecimal;  
import java.util.Objects;  
import java.util.Set;  
  
@DynamoDbImmutable(builder = ProductCatalog.Builder.class)
```

```
public class ProductCatalog implements Comparable<ProductCatalog> {  
    private Integer id;  
    private String title;  
    private String isbn;  
    private Set<String> authors;  
    private BigDecimal price;  
  
    private ProductCatalog(Builder builder){  
        this.authors = builder.authors;  
        this.id = builder.id;  
        this.isbn = builder.isbn;  
        this.price = builder.price;  
        this.title = builder.title;  
    }  
  
    public static Builder builder(){ return new Builder(); }  
  
    @DynamoDbPartitionKey  
    public Integer id() { return id; }  
  
    @DynamoDbSortKey  
    public String title() { return title; }  
  
    @DynamoDbSecondaryPartitionKey(indexNames = "products_by_isbn")  
    public String isbn() { return isbn; }  
    public Set<String> authors() { return authors; }  
    public BigDecimal price() { return price; }  
  
    public static final class Builder {  
        private Integer id;  
        private String title;  
        private String isbn;  
        private Set<String> authors;  
        private BigDecimal price;  
        private Builder(){}  
  
        public Builder id(Integer id) { this.id = id; return this; }  
        public Builder title(String title) { this.title = title; return this; }  
        public Builder isbn(String ISBN) { this.isbn = ISBN; return this; }  
        public Builder authors(Set<String> authors) { this.authors = authors; return  
this; }  
        public Builder price(BigDecimal price) { this.price = price; return this; }  
    }  
}
```

```
public ProductCatalog build() { return new ProductCatalog(this); }

}

@Override
public String toString() {
    final StringBuffer sb = new StringBuffer("ProductCatalog{");
    sb.append("id=").append(id);
    sb.append(", title='").append(title).append('\'');
    sb.append(", isbn='").append(isbn).append('\'');
    sb.append(", authors=").append(authors);
    sb.append(", price=").append(price);
    sb.append('}');
    return sb.toString();
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    ProductCatalog that = (ProductCatalog) o;
    return id.equals(that.id) && title.equals(that.title) && Objects.equals(isbn,
that.isbn) && Objects.equals(authors, that.authors) && Objects.equals(price,
that.price);
}

@Override
public int hashCode() {
    return Objects.hash(id, title, isbn, authors, price);
}

@Override
@DynamoDbIgnore
public int compareTo(ProductCatalog other) {
    if (this.id.compareTo(other.id) != 0){
        return this.id.compareTo(other.id);
    } else {
        return this.title.compareTo(other.title);
    }
}
}
```

Os dois exemplos de código a seguir de uma gravação em lote ilustram a verbosidade e a falta de segurança de tipos ao usar o cliente padrão em vez do cliente aprimorado.

Compare: Gravação em lote usando o cliente padrão do DynamoDB

```
public static void batchWriteStandard(DynamoDbClient dynamoDbClient, String tableName) {

    Map<String, AttributeValue> catalogItem = Map.of(
        "authors", AttributeValue.builder().ss("a", "b").build(),
        "id", AttributeValue.builder().n("1").build(),
        "isbn", AttributeValue.builder().s("1-565-85698").build(),
        "title", AttributeValue.builder().s("Title 1").build(),
        "price", AttributeValue.builder().n("52.13").build());

    Map<String, AttributeValue> catalogItem2 = Map.of(
        "authors", AttributeValue.builder().ss("a", "b", "c").build(),
        "id", AttributeValue.builder().n("2").build(),
        "isbn", AttributeValue.builder().s("1-208-98073").build(),
        "title", AttributeValue.builder().s("Title 2").build(),
        "price", AttributeValue.builder().n("21.99").build());

    Map<String, AttributeValue> catalogItem3 = Map.of(
        "authors", AttributeValue.builder().ss("g", "k", "c").build(),
        "id", AttributeValue.builder().n("3").build(),
        "isbn", AttributeValue.builder().s("7-236-98618").build(),
        "title", AttributeValue.builder().s("Title 3").build(),
        "price", AttributeValue.builder().n("42.00").build());

    Set<WriteRequest> writeRequests = Set.of(
        WriteRequest.builder().putRequest(b -> b.item(catalogItem)).build(),
        WriteRequest.builder().putRequest(b -> b.item(catalogItem2)).build(),
        WriteRequest.builder().putRequest(b -> b.item(catalogItem3)).build());

    Map<String, Set<WriteRequest>> productCatalogItems = Map.of(
        "ProductCatalog", writeRequests);

    BatchWriteItemResponse response = dynamoDbClient.batchWriteItem(b ->
        b.requestItems(productCatalogItems));

    logger.info("Unprocessed items: " + response.unprocessedItems().size());
}
```

Compare: Gravação em lote usando o DynamoDB Enhanced Client

```
public static void batchWriteEnhanced(DynamoDbTable<ProductCatalog> productCatalog)
{
    ProductCatalog prod = ProductCatalog.builder()
        .id(1)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(52.13))
        .title("Title 1")
        .build();
    ProductCatalog prod2 = ProductCatalog.builder()
        .id(2)
        .isbn("1-208-98073")
        .authors(new HashSet<>(Arrays.asList("a", "b", "c")))
        .price(BigDecimal.valueOf(21.99))
        .title("Title 2")
        .build();
    ProductCatalog prod3 = ProductCatalog.builder()
        .id(3)
        .isbn("7-236-98618")
        .authors(new HashSet<>(Arrays.asList("g", "k", "c")))
        .price(BigDecimal.valueOf(42.00))
        .title("Title 3")
        .build();

    BatchWriteResult batchWriteResult = DependencyFactory.enhancedClient()
        .batchWriteItem(b -> b.writeBatches(
            WriteBatch.builder(ProductCatalog.class)
                .mappedTableResource(productCatalog)
                .addPutItem(prod).addPutItem(prod2).addPutItem(prod3)
                .build()
        ));
    logger.info("Unprocessed items: " +
batchWriteResult.unprocessedPutItemsForTable(productCatalog).size());
}
```

Trabalhe com classes de dados imutáveis

O recurso de mapeamento da DynamoDB Enhanced Client API funciona com classes de dados imutáveis. Uma classe imutável tem apenas getters e requer uma classe construtora que o SDK usa para criar instâncias da classe. Em vez de usar a `@DynamoDbBean` anotação conforme mostrado

na [classe Customer](#), as classes imutáveis usam a `@DynamoDbImmutable` anotação, que usa um parâmetro que indica a classe do construtor a ser usada.

A classe a seguir é uma versão imutável de `Customer`

```
package org.example.tests.model.immutable;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbImmutable;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondaryPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.time.Instant;

@DynamoDbImmutable(builder = CustomerImmutable.Builder.class)
public class CustomerImmutable {
    private final String id;
    private final String name;
    private final String email;
    private final Instant regDate;

    private CustomerImmutable(Builder b) {
        this.id = b.id;
        this.email = b.email;
        this.name = b.name;
        this.regDate = b.regDate;
    }

    // This method will be automatically discovered and used by the TableSchema.
    public static Builder builder() { return new Builder(); }

    @DynamoDbPartitionKey
    public String id() { return this.id; }

    @DynamoDbSortKey
    public String email() { return this.email; }

    @DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name")
    public String name() { return this.name; }
```

```
@DynamoDbSecondarySortKey(indexNames = {"customers_by_date", "customers_by_name"})
public Instant regDate() { return this.regDate; }

public static final class Builder {
    private String id;
    private String email;
    private String name;
    private Instant regDate;

    // The private Builder constructor is visible to the enclosing Customer class.
    private Builder() {}

    public Builder id(String accountId) { this.id = id; return this; }
    public Builder email(String email) { this.email = email; return this; }
    public Builder name(String name) { this.name = name; return this; }
    public Builder regDate(Instant regDate) { this.regDate = regDate; return
this; }

    // This method will be automatically discovered and used by the TableSchema.
    public CustomerImmutable build() { return new CustomerImmutable(this); }
}
}
```

Você deve atender aos seguintes requisitos ao fazer anotações em uma classe de dados com `@DynamoDbImmutable`

1. Todo método que não é uma substituição `Object.class` e não foi anotado `@DynamoDbIgnore` deve ser um coletor de um atributo da tabela do DynamoDB.
2. Cada getter deve ter um setter correspondente com distinção entre maiúsculas e minúsculas na classe builder.
3. Somente uma das seguintes condições de construção deve ser atendida.
 - A classe builder deve ter um construtor padrão público.
 - A classe de dados deve ter um método estático público chamado `builder()` que não usa parâmetros e retorna uma instância da classe builder. Essa opção é mostrada na classe `imutávelCustomer`.
4. A classe builder deve ter um método público chamado `build()` que não use parâmetros e retorne uma instância da classe imutável.

Para criar um TableSchema para sua classe imutável, use o `fromImmutableClass()` método on `TableSchema` conforme mostrado no trecho a seguir.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =  
    TableSchema.fromImmutableClass(CustomerImmutable.class);
```

Assim como você pode criar uma tabela do DynamoDB a partir de uma classe mutável, você pode criar uma a partir de uma classe imutável com uma chamada única para `of`, conforme mostrado no exemplo `DynamoDbTable` de trecho `createTable()` a seguir.

```
static void createTableFromImmutable(DynamoDbEnhancedClient enhancedClient, String  
    tableName, DynamoDbWaiter waiter){  
    // First, create an in-memory representation of the table using the 'table()'  
    // method of the DynamoDb Enhanced Client.  
    // 'table()' accepts a name for the table and a TableSchema instance that you  
    // created previously.  
    DynamoDbTable<CustomerImmutable> customerDynamoDbTable = enhancedClient  
        .table(tableName, TableSchema.fromImmutableClass(CustomerImmutable.class));  
  
    // Second, call the 'createTable()' method on the DynamoDbTable instance.  
    customerDynamoDbTable.createTable();  
    waiter.waitUntilTableExists(b -> b.tableName(tableName));  
}
```

Use bibliotecas de terceiros, como Lombok

Bibliotecas de terceiros, como o [Project Lombok](#), ajudam a gerar código padronizado associado a objetos imutáveis. A API do DynamoDB Enhanced Client funciona com essas bibliotecas, desde que as classes de dados sigam as convenções detalhadas nesta seção.

O exemplo a seguir mostra a `CustomerImmutable` classe imutável com anotações do Lombok. Observe como o `onMethod` recurso do Lombok copia anotações do DynamoDB baseadas em atributos, como, no código gerado. `@DynamoDbPartitionKey`

```
@Value  
@Builder  
@DynamoDbImmutable(builder = Customer.CustomerBuilder.class)  
public class Customer {  
    @Getter(onMethod_=@DynamoDbPartitionKey)  
    private String id;
```

```
@Getter(onMethod_=@DynamoDbSortKey)
private String email;

@Getter(onMethod_=@DynamoDbSecondaryPartitionKey(indexNames = "customers_by_name"))
private String name;

@Getter(onMethod_=@DynamoDbSecondarySortKey(indexNames = {"customers_by_date",
"customers_by_name"}))
private Instant createdDate;
}
```

Expressões e condições

[As expressões na API do DynamoDB Enhanced Client são representações Java das expressões do DynamoDB.](#)

A API do DynamoDB Enhanced Client usa três tipos de expressões:

[Expressão](#)

A Expression classe é usada quando você define condições e filtros.

[QueryConditional](#)

Esse tipo de expressão representa [as principais condições](#) para operações de consulta.

[UpdateExpression](#)

Essa classe ajuda você a escrever expressões de [atualização do DynamoDB](#) e é usada atualmente na estrutura de extensão quando você atualiza um item.

Anatomia da expressão

Uma expressão é composta pelo seguinte:

- Uma expressão de string (obrigatória). A string contém uma expressão lógica do DynamoDB com nomes de espaço reservado para nomes e valores de atributos.
- Um mapa dos valores da expressão (geralmente obrigatório).
- Um mapa dos nomes das expressões (opcional).

Use um construtor para gerar um Expression objeto que tenha a seguinte forma geral.

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

Expressions geralmente exigem um mapa dos valores da expressão. O mapa fornece os valores para os espaços reservados na expressão de seqüência de caracteres. A chave do mapa consiste no nome do espaço reservado precedido por dois pontos (:) e o valor do mapa é uma instância de [AttributeValue](#). A [AttributeValue](#) classe tem métodos convenientes para gerar uma [AttributeValue](#) instância a partir de um literal. Como alternativa, você pode usar o [AttributeValue.Builder](#) para gerar uma [AttributeValue](#) instância.

O trecho a seguir mostra um mapa com duas entradas após a linha de comentário 2. A string passada para o `expression()` método, mostrada após a linha de comentário 1, contém os espaços reservados que o DynamoDB resolve antes de realizar a operação. Esse trecho não contém um mapa dos nomes das expressões, porque o preço é um nome de atributo permitido.

```
public static void scanAsync(DynamoDbAsyncTable productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();
```

Se um nome de atributo na tabela do DynamoDB for uma palavra reservada, começar com um número ou contiver um espaço, um mapa dos nomes das expressões será necessário para o `Expression`.

Por exemplo, se o nome do atributo fosse `price` em `1price` vez do exemplo de código anterior, o exemplo precisaria ser modificado conforme mostrado no exemplo a seguir.

```
ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .filterExpression(Expression.builder()
        .expression("#price >= :min_value AND #price <= :max_value")
        .expressionNames( Map.of("#price", "1price") )
        .expressionValues(
            Map.of(":min_value", numberValue(8.00),
                  ":max_value", numberValue(400_000.00)))
        .build())
    .build();
```

Um espaço reservado para o nome de uma expressão começa com o sinal de libra (#). Uma entrada para o mapa de nomes de expressão usa o espaço reservado como chave e o nome do atributo como valor. O mapa é adicionado ao construtor de expressões com o `expressionNames()` método. O DynamoDB resolve o nome do atributo antes de realizar a operação.

Os valores de expressão não são necessários se uma função for usada na expressão de cadeia de caracteres. Um exemplo de função de expressão é `attribute_exists(<attribute_name>)`.

O exemplo a seguir cria um `Expression` que usa uma função do [DynamoDB](#). A string de expressão neste exemplo não usa espaços reservados. Essa expressão pode ser usada em uma `putItem` operação para verificar se um item já existe no banco de dados com o valor de um `movie` atributo igual ao `movie` atributo do objeto de dados.

```
Expression exp = Expression.builder().expression("attribute_not_exists
(movie)").build();
```

O [DynamoDB Developer Guide](#) contém informações completas sobre as [expressões de baixo nível usadas com o DynamoDB](#).

Expressões condicionais e condicionais

Ao usar os `deleteItem()` método, `putItem()`, `updateItem()`, e, e também ao usar operações de transação e em lote, você usa [Expression](#) objetos para especificar as condições que o DynamoDB deve atender para continuar com a operação. Essas expressões são chamadas de expressões condicionais. Para ver um exemplo, consulte a expressão de condição usada no `addDeleteItem()` método (após a linha de comentário 1) do [exemplo de transação](#) mostrado neste guia.

Quando você trabalha com os `query()` métodos, uma condição é expressa como [QueryConditional](#). A `QueryConditional` classe tem vários métodos estáticos de conveniência que ajudam você a escrever os critérios que determinam quais itens ler no DynamoDB.

Para ver exemplos de `QueryConditionals`, consulte o primeiro exemplo de código da [the section called “Query exemplos de métodos”](#) seção deste guia.

Expressões de filtro

As expressões de filtro são usadas em operações de digitalização e consulta para filtrar os itens retornados.

Uma expressão de filtro é aplicada depois que todos os dados são lidos do banco de dados, então o custo de leitura é o mesmo que se não houvesse filtro. [O Amazon DynamoDB Developer Guide tem mais informações sobre o uso de expressões de filtro para operações de consulta e verificação.](#)

O exemplo a seguir mostra uma expressão de filtro adicionada a uma solicitação de escaneamento. O critério restringe os itens devolvidos a itens com um preço entre 8,00 e 80,00, inclusive.

```
Map<String, AttributeValue> expressionValues = Map.of(
    ":min_value", numberValue(8.00),
    ":max_value", numberValue(80.00));

ScanEnhancedRequest request = ScanEnhancedRequest.builder()
    .consistentRead(true)
    // 1. the 'attributesToProject()' method allows you to specify which
values you want returned.
    .attributesToProject("id", "title", "authors", "price")
    // 2. Filter expression limits the items returned that match the
provided criteria.
    .filterExpression(Expression.builder()
        .expression("price >= :min_value AND price <= :max_value")
        .expressionValues(expressionValues)
        .build())
    .build();
```

Expressões de atualização

O método do DynamoDB Enhanced Client fornece uma forma padrão `updateItem()` de atualizar itens no DynamoDB. [No entanto, quando você precisar de mais funcionalidades, UpdateExpressions forneça uma representação segura da sintaxe da expressão de atualização do DynamoDB.](#) Por exemplo, você pode usar `UpdateExpressions` para aumentar valores sem

primeiro ler itens do DynamoDB ou adicionar membros individuais a uma lista. Atualmente, as expressões de atualização estão disponíveis em extensões personalizadas para o `updateItem()` método.

Para ver um exemplo que usa expressões de atualização, consulte o [exemplo de extensão personalizada](#) neste guia.

Mais informações sobre expressões de atualização estão disponíveis no [Amazon DynamoDB Developer Guide](#).

Trabalhe com vários itens

Os `scan` e `batch` métodos `query` e da API DynamoDB Enhanced Client retornam respostas com uma ou mais páginas. Uma página contém um ou mais itens. Seu código pode processar a resposta por página ou pode processar itens individuais.

Uma resposta paginada retornada pelo `DynamoDbEnhancedClient` cliente síncrono retorna um [PageIterable](#) objeto, enquanto uma resposta retornada pelo `DynamoDbEnhancedAsyncClient` assíncrono retorna um objeto. [PagePublisher](#)

Esta seção analisa o processamento de resultados paginados e fornece exemplos que usam as APIs de verificação e consulta.

Tópicos

- [Verificar uma tabela](#)
- [Consultar uma tabela](#)

Verificar uma tabela

O `scan` método do SDK corresponde à operação do [DynamoDB com o mesmo nome](#). A API do DynamoDB Enhanced Client oferece as mesmas opções, mas usa um modelo de objeto familiar e gerencia a paginação para você.

Primeiro, exploramos a `PageIterable` interface examinando o `scan` método da classe de mapeamento síncrono, [DynamoDbTable](#).

Use a API síncrona

O exemplo a seguir mostra o `scan` método que usa uma [expressão](#) para filtrar os itens retornados. [ProductCatalog](#) É o objeto do modelo que foi mostrado anteriormente.

A expressão de filtragem mostrada após a linha de comentário 1 limita os ProductCatalog itens que são retornados àqueles com um valor de preço entre 8,00 e 80,00, inclusive.

Este exemplo também exclui os isbn valores usando o attributesToProject método mostrado após a linha de comentário 2.

Na linha de comentário 3, o PageIterable objeto,pagedResult, é retornado pelo scan método. O stream método de PageIterable retorna um [java.util.Stream](#) objeto, que você pode usar para processar as páginas. Neste exemplo, o número de páginas é contado e registrado.

Começando com a linha de comentários 4, o exemplo mostra duas variações de acesso aos ProductCatalog itens. A versão após a linha de comentário 2a percorre cada página e classifica e registra os itens em cada página. A versão após a linha de comentário 2b ignora a iteração da página e acessa os itens diretamente.

A PageIterable interface oferece várias maneiras de processar resultados por causa de suas duas interfaces principais — [java.lang.Iterable](#)[SdkIterable](#). Iterable traz o `forEach`, `iterator` e `spliterator` métodos, e SdkIterable traz o `stream` método.

```
public static void scanSync(DynamoDbTable<ProductCatalog> productCatalog) {  
  
    Map<String, AttributeValue> expressionValues = Map.of(  
        ":min_value", numberValue(8.00),  
        ":max_value", numberValue(80.00));  
  
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()  
        .consistentRead(true)  
        // 1. the 'attributesToProject()' method allows you to specify which  
        values you want returned.  
        .attributesToProject("id", "title", "authors", "price")  
        // 2. Filter expression limits the items returned that match the  
        provided criteria.  
        .filterExpression(Expression.builder()  
            .expression("price >= :min_value AND price <= :max_value")  
            .expressionValues(expressionValues)  
            .build())  
        .build();  
  
    // 3. A PageIterable object is returned by the scan method.  
    PageIterable<ProductCatalog> pagedResults = productCatalog.scan(request);  
    logger.info("page count: {}", pagedResults.stream().count());
```

```
// 4. Log the returned ProductCatalog items using two variations.  
// 4a. This version sorts and logs the items of each page.  
pagedResults.stream().forEach(p -> p.items().stream()  
    .sorted(Comparator.comparing(ProductCatalog::price))  
    .forEach(  
        item -> logger.info(item.toString())  
    ));  
// 4b. This version sorts and logs all items for all pages.  
pagedResults.items().stream()  
    .sorted(Comparator.comparing(ProductCatalog::price))  
    .forEach(  
        item -> logger.info(item.toString())  
    );  
}
```

Use a API assíncrona

O `scan` método assíncrono retorna os resultados como um objeto `PagePublisher`. A `PagePublisher` interface tem dois `subscribe` métodos que você pode usar para processar páginas de resposta. Um `subscribe` método vem da interface `org.reactivestreams.Publisher` principal. Para processar páginas usando essa primeira opção, transmita uma [Subscriber](#) instância para o `subscribe` método. O primeiro exemplo a seguir mostra o uso do `subscribe` método.

O segundo `subscribe` método vem da [SdkPublisher](#) interface. Esta versão do `subscribe` aceita um [Consumer](#) em vez de um `Subscriber`. Essa variação do `subscribe` método é mostrada no segundo exemplo a seguir.

O exemplo a seguir mostra a versão assíncrona do `scan` método que usa a mesma expressão de filtro mostrada no exemplo anterior.

Após a linha de comentário 3, `DynamoDbAsyncTable.scan` retorna um `PagePublisher` objeto. Na próxima linha, o código cria uma instância da `org.reactivestreams.Subscriber` interface, `ProductCatalogSubscriber`, que se inscreve na linha 4 `PagePublisher` após o comentário.

O `Subscriber` objeto coleta os `ProductCatalog` itens de cada página no `onNext` método após a linha de comentário 8 no exemplo da `ProductCatalogSubscriber` classe. Os itens são armazenados na `List` variável privada e acessados no código de chamada com o `ProductCatalogSubscriber.getSubscribedItems()` método. Isso é chamado após a linha de comentários 5.

Depois que a lista é recuperada, o código classifica todos os ProductCatalog itens por preço e registra cada item.

O [CountDownLatch](#)in the ProductCatalogSubscriber class bloqueia o tópico de chamada até que todos os itens tenham sido adicionados à lista antes de continuar após a linha de comentários 5.

```
public static void scanAsync(DynamoDbAsyncTable<ProductCatalog> productCatalog) {
    ScanEnhancedRequest request = ScanEnhancedRequest.builder()
        .consistentRead(true)
        .attributesToProject("id", "title", "authors", "price")
        .filterExpression(Expression.builder()
            // 1. :min_value and :max_value are placeholders for the values
            provided by the map
            .expression("price >= :min_value AND price <= :max_value")
            // 2. Two values are needed for the expression and each is
            supplied as a map entry.
            .expressionValues(
                Map.of( ":min_value", numberValue(8.00),
                    ":max_value", numberValue(400_000.00)))
            .build())
        .build();

    // 3. A PagePublisher object is returned by the scan method.
    PagePublisher<ProductCatalog> pagePublisher = productCatalog.scan(request);
    ProductCatalogSubscriber subscriber = new ProductCatalogSubscriber();
    // 4. Subscribe the ProductCatalogSubscriber to the PagePublisher.
    pagePublisher.subscribe(subscriber);
    // 5. Retrieve all collected ProductCatalog items accumulated by the
    subscriber.
    subscriber.getSubscribedItems().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString()));
    // 6. Use a Consumer to work through each page.
    pagePublisher.subscribe(page -> page
        .items().stream()
        .sorted(Comparator.comparing(ProductCatalog::price))
        .forEach(item ->
            logger.info(item.toString())))
    .join(); // If needed, blocks the subscribe() method thread until it is
    finished processing.
    // 7. Use a Consumer to work through each ProductCatalog item.
    pagePublisher.items()
```

```
.subscribe(product -> logger.info(product.toString()))
.exceptionally(failure -> {
    logger.error("ERROR - ", failure);
    return null;
})
.join(); // If needed, blocks the subscribe() method thread until it is
finished processing.
}
```

```
private static class ProductCatalogSubscriber implements
Subscriber<Page<ProductCatalog>> {
    private CountDownLatch latch = new CountDownLatch(1);
    private Subscription subscription;
    private List<ProductCatalog> itemsFromAllPages = new ArrayList<>();

    @Override
    public void onSubscribe(Subscription sub) {
        subscription = sub;
        subscription.request(1L);
        try {
            latch.await(); // Called by main thread blocking it until latch is
released.
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onNext(Page<ProductCatalog> productCatalogPage) {
        // 8. Collect all the ProductCatalog instances in the page, then ask the
publisher for one more page.
        itemsFromAllPages.addAll(productCatalogPage.items());
        subscription.request(1L);
    }

    @Override
    public void onError(Throwable throwable) {
    }

    @Override
    public void onComplete() {
        latch.countDown(); // Call by subscription thread; latch releases.
    }
}
```

```
List<ProductCatalog> getSubscribedItems() {  
    return this.itemsFromAllPages;  
}  
}
```

O exemplo de trecho a seguir usa a versão do `PagePublisher.subscribe` método que aceita a linha 6 `Consumer` após o comentário. O parâmetro lambda Java consome páginas, que processam ainda mais cada item. Neste exemplo, cada página é processada e os itens em cada página são classificados e, em seguida, registrados.

```
// 6. Use a Consumer to work through each page.  
pagePublisher.subscribe(page -> page  
        .items().stream()  
        .sorted(Comparator.comparing(ProductCatalog::price))  
        .forEach(item ->  
            logger.info(item.toString())))  
    .join(); // If needed, blocks the subscribe() method thread until it is  
finished processing.
```

O `items` método de `PagePublisher` desempacotar as instâncias do modelo para que seu código possa processar os itens diretamente. Essa abordagem é mostrada no trecho a seguir.

```
// 7. Use a Consumer to work through each ProductCatalog item.  
pagePublisher.items()  
    .subscribe(product -> logger.info(product.toString()))  
    .exceptionally(failure -> {  
        logger.error("ERROR - ", failure);  
        return null;  
    })  
    .join(); // If needed, blocks the subscribe() method thread until it is  
finished processing.
```

Consultar uma tabela

O [query\(\)](#) método da `DynamoDbTable` classe encontra itens com base nos valores da chave primária. A `@DynamoDbPartitionKey` anotação e a `@DynamoDbSortKey` anotação opcional são usadas para definir a chave primária em sua classe de dados.

O `query()` método requer um valor de chave de partição que encontre itens que correspondam ao valor fornecido. Se sua tabela também definir uma chave de classificação, você poderá adicionar

um valor para ela à sua consulta como uma condição de comparação adicional para ajustar os resultados.

Exceto pelo processamento dos resultados, as versões síncrona e assíncrona do funcionam da `query()` mesma forma. Assim como na `scan` API, a `query` API retorna um `PageIterable` para uma chamada síncrona e um `PagePublisher` para uma chamada assíncrona. Discutimos o uso de `PageIterable` e `PagePublisher` anteriormente na seção de digitalização.

Queryexemplos de métodos

O exemplo de código do `query()` método a seguir usa a `MovieActor` classe. A classe de dados define uma chave primária composta composta pelo `movie`atributo da chave de partição e pelo `actor`atributo da chave de classificação.

A classe também sinaliza que usa um índice secundário global chamado `acting_award_year`. A chave primária composta do índice é composta pelo `actingaward`atributo da chave de partição e pelo atributo da `actingyear`chave de classificação. Posteriormente neste tópico, quando mostrarmos como criar e usar índices, nos referiremos ao `acting_award_year`índice.

Classe MovieActor

```
package org.example.tests.model;

import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbAttribute;
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.DynamoDbSecondaryPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.Objects;

@DynamoDbBean
public class MovieActor implements Comparable<MovieActor> {

    private String movieName;
    private String actorName;
    private String actingAward;
    private Integer actingYear;
```

```
private String actingSchoolName;

@DynamoDbPartitionKey
@DynamoDbAttribute("movie")
public String getMovieName() {
    return movieName;
}

public void setMovieName(String movieName) {
    this.movieName = movieName;
}

@DynamoDbSortKey
@DynamoDbAttribute("actor")
public String getActorName() {
    return actorName;
}

public void setActorName(String actorName) {
    this.actorName = actorName;
}

@DynamoDbSecondaryPartitionKey(indexNames = "acting_award_year")
@DynamoDbAttribute("actingaward")
public String getActingAward() {
    return actingAward;
}

public void setActingAward(String actingAward) {
    this.actingschoolname = actingAward;
}

@DynamoDbSecondarySortKey(indexNames = {"acting_award_year", "movie_year"})
@DynamoDbAttribute("actingyear")
public Integer getActingYear() {
    return actingYear;
}

public void setActingYear(Integer actingYear) {
    this.actingschoolname = actingYear;
}

@DynamoDbAttribute("actingschoolname")
public String getActingSchoolName() {
```

```
        return actingSchoolName;
    }

    public void setActingSchoolName(String actingSchoolName) {
        this.actingschoolName = actingSchoolName;
    }

    @Override
    public String toString() {
        final StringBuffer sb = new StringBuffer("MovieActor{");
        sb.append("movieName='").append(movieName).append('\'');
        sb.append(", actorName='").append(actorName).append('\'');
        sb.append(", actingAward='").append(actingAward).append('\'');
        sb.append(", actingYear=").append(actingYear);
        sb.append(", actingSchoolName='").append(actingSchoolName).append('\'');
        sb.append('}');
        return sb.toString();
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        MovieActor that = (MovieActor) o;
        return Objects.equals(movieName, that.movieName) && Objects.equals(actorName,
that.actorName) && Objects.equals(actingAward, that.actingAward) &&
Objects.equals(actingYear, that.actingYear) && Objects.equals(actingSchoolName,
that.actingSchoolName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(movieName, actorName, actingAward, actingYear,
actingSchoolName);
    }

    @Override
    public int compareTo(MovieActor o) {
        if (this.movieName.compareTo(o.movieName) != 0){
            return this.movieName.compareTo(o.movieName);
        } else {
            return this.actorName.compareTo(o.actorName);
        }
    }
}
```

}

Os exemplos de código a seguir são consultados com base nos itens a seguir.

Itens na **MovieActor** tabela

```
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',  
actingYear=2001, actingSchoolName='null'}  
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',  
actingYear=2001, actingSchoolName='actingschool1'}  
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',  
actingYear=2001, actingSchoolName='actingschool2'}  
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',  
actingYear=2001, actingSchoolName='null'}  
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',  
actingYear=2001, actingSchoolName='actingschool4'}  
MovieActor{movieName='movie02', actorName='actor0', actingAward='actingaward0',  
actingYear=2002, actingSchoolName='null'}  
MovieActor{movieName='movie02', actorName='actor1', actingAward='actingaward1',  
actingYear=2002, actingSchoolName='actingschool1'}  
MovieActor{movieName='movie02', actorName='actor2', actingAward='actingaward2',  
actingYear=2002, actingSchoolName='actingschool2'}  
MovieActor{movieName='movie02', actorName='actor3', actingAward='actingaward3',  
actingYear=2002, actingSchoolName='null'}  
MovieActor{movieName='movie02', actorName='actor4', actingAward='actingaward4',  
actingYear=2002, actingSchoolName='actingschool4'}  
MovieActor{movieName='movie03', actorName='actor0', actingAward='actingaward0',  
actingYear=2003, actingSchoolName='null'}  
MovieActor{movieName='movie03', actorName='actor1', actingAward='actingaward1',  
actingYear=2003, actingSchoolName='actingschool1'}  
MovieActor{movieName='movie03', actorName='actor2', actingAward='actingaward2',  
actingYear=2003, actingSchoolName='actingschool2'}  
MovieActor{movieName='movie03', actorName='actor3', actingAward='actingaward3',  
actingYear=2003, actingSchoolName='null'}  
MovieActor{movieName='movie03', actorName='actor4', actingAward='actingaward4',  
actingYear=2003, actingSchoolName='actingschool4'}
```

O código a seguir define duas [QueryConditional](#) instâncias. [QueryConditional](#)s trabalham com valores de chave, seja a chave de partição isolada ou em combinação com a chave de classificação, e correspondem às [principais expressões condicionais da API de serviço do DynamoDB](#). Depois da linha de comentário 1, o exemplo define a keyEqual instância que corresponde aos itens com um valor de partição de **movie01**.

Este exemplo também define uma expressão de filtro que filtra qualquer item que não esteja `actingschoolname` ligado após a linha de comentário 2.

Depois da linha de comentário 3, o exemplo mostra a `QueryEnhancedRequest` instância que o código passa para o `DynamoDbTable.query()` método. Esse objeto combina a condição principal e o filtro que o SDK usa para gerar a solicitação para o serviço do DynamoDB.

```
public static void query(DynamoDbTable movieActorTable) {  
  
    // 1. Define a QueryConditional instance to return items matching a partition  
    // value.  
    QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->  
        b.partitionValue("movie01"));  
    // 1a. Define a QueryConditional that adds a sort key criteria to the partition  
    // value criteria.  
    QueryConditional sortGreaterThanOrEqualTo =  
        QueryConditional.sortGreaterThanOrEqualTo(b ->  
            b.partitionValue("movie01").sortValue("actor2"));  
    // 2. Define a filter expression that filters out items whose attribute value  
    // is null.  
    final Expression filterOutNoActingschoolname =  
        Expression.builder().expression("attribute_exists(actingschoolname)").build();  
  
    // 3. Build the query request.  
    QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()  
        .queryConditional(keyEqual)  
        .filterExpression(filterOutNoActingschoolname)  
        .build();  
    // 4. Perform the query.  
    PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);  
    logger.info("page count: {}", pagedResults.stream().count()); // Log number of  
    // pages.  
  
    pagedResults.items().stream()  
        .sorted()  
        .forEach(  
            item -> logger.info(item.toString()) // Log the sorted list of  
            // items.  
        );  
}
```

A seguir está o resultado da execução do método. A saída exibe itens com um `movieName` valor de `movie01` e não exibe nenhum item `actingSchoolName` igual a `null`.

```
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
actingYear=2001, actingSchoolName='actingschool1'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:11:05 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
actingYear=2001, actingSchoolName='actingschool4'}
```

Na seguinte variação de solicitação de consulta mostrada anteriormente após a linha de comentário 3, o código substitui o pelo `keyEqual` `QueryConditional` `sortGreaterThanOrEqualTo` `QueryConditional` que foi definido após a linha de comentário 1a. O código a seguir também remove a expressão do filtro.

```
QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(sortGreaterThanOrEqualTo)
```

Como essa tabela tem uma chave primária composta, todas as `QueryConditional` instâncias exigem um valor de chave de partição. `QueryConditional` métodos que começam com `sort...` indicam que uma chave de classificação é necessária. Os resultados não são classificados.

A saída a seguir exibe os resultados da consulta. A consulta retorna itens que têm um `movieName` valor igual a filme01 e somente itens que têm um `actorName` valor maior ou igual a ator2. Como o filtro foi removido, a consulta retorna itens que não têm valor para o `actingSchoolName` atributo.

```
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:46 - page count: 1
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
actingYear=2001, actingSchoolName='actingschool2'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',
actingYear=2001, actingSchoolName='null'}
2023-03-05 13:15:00 [main] INFO org.example.tests.QueryDemo:51 -
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',
actingYear=2001, actingSchoolName='actingschool4'}
```

Várias operações

A API do DynamoDB Enhanced Client oferece suporte a várias opções por solicitação com lotes e transações. A diferença entre elas é que uma transação falha se uma ou mais operações individuais falharem, mas uma solicitação em lote permite que as operações individuais falhem enquanto processam as outras.

Sumário

- [Operações em lote](#)
 - [Exemplo de batchGetItem\(\)](#)
 - [Exemplo de batchWriteItem\(\)](#)
- [Operações de transação](#)
 - [Exemplo de transactGetItems\(\)](#)
 - [Exemplos do transactWriteItems\(\)](#)
 - [Exemplo básico](#)
 - [Exemplo de verificação de condição](#)
 - [Exemplo de condição de operação única](#)

Operações em lote

[A API do DynamoDB Enhanced Client oferece dois métodos em lote batchGetItem, \(\) e \(\)batchWriteItem.](#)

Exemplo de **batchGetItem()**

Com o [DynamoDbTable.batchGetItem\(\)](#) método, você pode recuperar até 100 itens individuais em várias tabelas em uma solicitação geral. O exemplo a seguir usa as classes de [MovieActor](#) dados [Customer](#) e mostradas anteriormente.

No exemplo após as linhas 1 e 2, você cria [ReadBatch](#) objetos que são adicionados posteriormente como parâmetros ao `batchGetItem()` método após a linha de comentário 3. O código após a linha de comentário 1 cria o lote para ser lido na `Customer` tabela. O código após a linha de comentário 1a mostra o uso de um [GetItemEnhancedRequest](#) construtor que usa valores de chave primária para especificar o item a ser lido. Ao contrário de especificar valores-chave para solicitar um item, você pode usar uma classe de dados para solicitar um item, conforme mostrado após a linha de comentário 1b. O SDK extrai os valores-chave nos bastidores antes de enviar a solicitação.

Ao especificar o item usando a abordagem baseada em chaves, conforme mostrado nas duas instruções apóis 2a, você também pode especificar que o DynamoDB deve realizar uma leitura altamente consistente. Quando o `consistentRead()` método é usado, ele deve ser usado em todos os itens solicitados para a mesma tabela.

Para recuperar os itens encontrados pelo DynamoDB, use `resultsForTable()` o método mostrado apóis a linha de comentário 4. Chame o método para cada tabela que foi lida na solicitação. `resultsForTable()` retorna uma lista de itens encontrados que você pode processar usando qualquer `java.util.List` método. Este exemplo registra cada item.

Para descobrir itens que o DynamoDB não processou, use a abordagem apóis a linha de comentários 5. A `BatchGetResultPage` classe tem o `unprocessKeysForTable()` método que dá acesso a cada chave que não foi processada. A [referência BatchGetItem da API](#) tem mais informações sobre situações que resultam em itens não processados.

```
public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                         DynamoDbTable<Customer> customerTable,
                                         DynamoDbTable<MovieActor> movieActorTable) {

    Customer customer2 = new Customer();
    customer2.setId("2");
    customer2.setEmail("cust2@example.org");

    // 1. Build a batch to read from the Customer table.
    ReadBatch customerBatch = ReadBatch.builder(Customer.class)
        .mappedTableResource(customerTable)
        // 1a. Specify the primary key values for the item.
        .add.GetItem(b -> b.key(k ->
    k.partitionValue("1").sortValue("cust1@orgname.org")))
        // 1b. Alternatively, supply a data class instances to provide the
    primary key values.
        .add.GetItem(customer2)
        .build();

    // 2. Build a batch to read from the MovieActor table.
    ReadBatch moveActorBatch = ReadBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
        // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
    table.
        .add.GetItem(b -> b.key(k ->
    k.partitionValue("movie01").sortValue("actor1")).consistentRead(Boolean.TRUE))
```

```
        .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")).consistentRead(Boolean.TRUE))
        .build();

    // 3. Add ReadBatch objects to the request.
    BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
b.readBatches(customerBatch, moveActorBatch));

    // 4. Retrieve the successfully requested items from each table.
    resultPages.resultsForTable(customerTable).forEach(item ->
logger.info(item.toString()));
    resultPages.resultsForTable(movieActorTable).forEach(item ->
logger.info(item.toString()));

    // 5. Retrieve the keys of the items requested but not processed by the
service.
    resultPages.forEach((BatchGetResultPage pageResult) -> {
        pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
        pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
    });
}
```

Suponha que os itens a seguir estejam nas duas tabelas antes de executar o código de exemplo.

Itens em tabelas

```
Customer [id=1, name=CustName1, email=cust1@example.org,
regDate=2023-03-31T15:46:27.688Z]
Customer [id=2, name=CustName2, email=cust2@example.org,
regDate=2023-03-31T15:46:28.688Z]
Customer [id=3, name=CustName3, email=cust3@example.org,
regDate=2023-03-31T15:46:29.688Z]
Customer [id=4, name=CustName4, email=cust4@example.org,
regDate=2023-03-31T15:46:30.688Z]
Customer [id=5, name=CustName5, email=cust5@example.org,
regDate=2023-03-31T15:46:31.689Z]
MovieActor{movieName='movie01', actorName='actor0', actingAward='actingaward0',
actingYear=2001, actingSchoolName='null'}
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',
actingYear=2001, actingSchoolName='actingschool1'}
MovieActor{movieName='movie01', actorName='actor2', actingAward='actingaward2',
actingYear=2001, actingSchoolName='actingschool2'}
```

```
MovieActor{movieName='movie01', actorName='actor3', actingAward='actingaward3',  
actingYear=2001, actingSchoolName='null'}  
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',  
actingYear=2001, actingSchoolName='actingschool4'}
```

A saída a seguir mostra os itens retornados e registrados após a linha de comentários 4.

```
Customer [id=1, name=CustName1, email=cust1@example.org,  
regDate=2023-03-31T15:46:27.688Z]  
Customer [id=2, name=CustName2, email=cust2@example.org,  
regDate=2023-03-31T15:46:28.688Z]  
MovieActor{movieName='movie01', actorName='actor4', actingAward='actingaward4',  
actingYear=2001, actingSchoolName='actingschool4'}  
MovieActor{movieName='movie01', actorName='actor1', actingAward='actingaward1',  
actingYear=2001, actingSchoolName='actingschool1'}
```

Exemplo de **batchWriteItem()**

O `batchWriteItem()` método coloca ou exclui vários itens em uma ou mais tabelas. Você pode especificar até 25 operações individuais de colocação ou exclusão na solicitação. O exemplo a seguir usa as classes [ProductCatalog](#) [MovieActor](#) modelo mostradas anteriormente.

`WriteBatch`s objetos são construídos após as linhas de comentário 1 e 2. Para a `ProductCatalog` tabela, o código coloca um item e exclui um item. Para a `MovieActor` tabela após a linha de comentário 2, o código coloca dois itens e exclui um.

O `batchWriteItem` método é chamado após a linha de comentário 3. O [builder](#) parâmetro fornece as solicitações em lote para cada tabela.

O [BatchWriteResult](#) objeto retornado fornece métodos separados para cada operação para visualizar solicitações não processadas. O código após a linha de comentário 4a fornece as chaves para solicitações de exclusão não processadas e o código após a linha de comentário 4b fornece os itens de venda não processados.

```
public static void batchWriteItemExample(DynamoDbEnhancedClient enhancedClient,  
                                         DynamoDbTable<ProductCatalog>  
                                         catalogTable,  
                                         DynamoDbTable<MovieActor> movieActorTable)  
{  
  
    // 1. Build a batch to write to the ProductCatalog table.  
    WriteBatch products = WriteBatch.builder(ProductCatalog.class)
```

```

        .mappedTableResource(catalogTable)
        .addPutItem(b -> b.item(getProductCatItem1()))
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getProductCatItem2().id())
            .sortValue(getProductCatItem2().title())))
        .build();

    // 2. Build a batch to write to the MovieActor table.
    WriteBatch movies = WriteBatch.builder(MovieActor.class)
        .mappedTableResource(movieActorTable)
        .addPutItem(getMovieActorYeoh())
        .addPutItem(getMovieActorBlanchettPartial())
        .addDeleteItem(b -> b.key(k -> k
            .partitionValue(getMovieActorStreep().getMovieName())
            .sortValue(getMovieActorStreep().getActorName())))
        .build();

    // 3. Add WriteBatch objects to the request.
    BatchWriteResult batchWriteResult = enhancedClient.batchWriteItem(b ->
    b.writeBatches(products, movies));
    // 4. Retrieve keys for items the service did not process.
    // 4a. 'unprocessedDeleteItemsForTable()' returns keys for delete requests that
    did not process.
    if (batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).size() >
    0) {

batchWriteResult.unprocessedDeleteItemsForTable(movieActorTable).forEach(key ->
    logger.info(key.toString()));
}
// 4b. 'unprocessedPutItemsForTable()' returns keys for put requests that did
not process.
if (batchWriteResult.unprocessedPutItemsForTable(catalogTable).size() > 0) {
    batchWriteResult.unprocessedPutItemsForTable(catalogTable).forEach(key ->
        logger.info(key.toString()));
}
}
}

```

Os métodos auxiliares a seguir fornecem os objetos de modelo para as operações put e delete.

Métodos auxiliares

```

public static ProductCatalog getProductCatItem1() {
    return ProductCatalog.builder()
        .id(2)

```

```
.isbn("1-565-85698")
.authors(new HashSet<>(Arrays.asList("a", "b")))
.price(BigDecimal.valueOf(30.22))
.title("Title 55")
.build();
}

public static ProductCatalog getProductCatItem2() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchettPartial() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    return movieActor;
}

public static MovieActor getMovieActorStreep() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Meryl Streep");
    movieActor.setMovieName("Sophie's Choice");
    movieActor.setActingYear(1982);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Yale School of Drama");
    return movieActor;
}

public static MovieActor getMovieActorYeoh(){
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Michelle Yeoh");
    movieActor.setMovieName("Everything Everywhere All at Once");
    movieActor.setActingYear(2023);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("Royal Academy of Dance");
    return movieActor;
}
```

Suponha que as tabelas contenham os itens a seguir antes de você executar o código de exemplo.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}  
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}  
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
```

Depois que o código de exemplo for concluído, as tabelas conterão os itens a seguir.

```
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best Actress', actingYear=2013, actingSchoolName='null'}  
MovieActor{movieName='Everything Everywhere All at Once', actorName='Michelle Yeoh', actingAward='Best Actress', actingYear=2023, actingSchoolName='Royal Academy of Dance'}  
ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b], price=30.22}
```

Observe na MovieActor tabela que o item do Blue Jasmine filme foi substituído pelo item usado na solicitação obtida por meio do método `getMovieActorBlanchettPartial()` auxiliar. Se um valor de atributo do bean de dados não tiver sido fornecido, o valor no banco de dados será removido. É por isso que o resultado `actingSchoolName` é nulo para o item do Blue Jasmine filme.

Note

Embora a documentação da API sugira que expressões de condição possam ser usadas e que a capacidade consumida e as métricas de coleta possam ser retornadas com solicitações individuais de [colocação](#) e [exclusão](#), esse não é o caso em um cenário de gravação em lote. Para melhorar o desempenho das operações em lote, essas opções individuais são ignoradas.

Operações de transação

A API do DynamoDB Enhanced Client fornece `transactGetItems()` e os métodos `transactWriteItems()`. Os métodos de transação do SDK for Java fornecem atomicidade, consistência, isolamento e durabilidade (ACID) nas tabelas do DynamoDB, ajudando você a manter a exatidão dos dados em seus aplicativos.

Sumário

- [Exemplo de transactGetItems\(\)](#)
- [Exemplos do transactWriteItems\(\)](#)
 - [Exemplo básico](#)
 - [Exemplo de verificação de condição](#)
 - [Exemplo de condição de operação única](#)

Exemplo de `transactGetItems()`

O `transactGetItems()` método aceita até 100 solicitações individuais de itens. Todos os itens são lidos em uma única transação atômica. O Amazon DynamoDB Developer Guide tem informações sobre [as condições que causam a falha de `transactGetItems\(\)` um método](#) e também sobre o nível de isolamento usado quando você liga. [`transactGetItem\(\)`](#)

Depois da linha de comentário 1 no exemplo a seguir, o código chama o `transactGetItems()` método com um `builder` parâmetro. O do construtor `addGetItem()` é invocado três vezes com um objeto de dados que contém os valores-chave que o SDK usará para gerar a solicitação final.

A solicitação retorna uma lista de `Document` objetos após a linha de comentário 2. A lista de documentos retornada contém instâncias de `documentos` não nulas dos dados do item na mesma ordem solicitada. O `Document.getItem(MappedTableResource<T> mappedTableResource)` método converte um objeto não digitado em um `Document` objeto Java digitado se os dados do item forem retornados, caso contrário, o método retornará null.

```
public static void transactGetItemsExample(DynamoDbEnhancedClient enhancedClient,
                                            DynamoDbTable<ProductCatalog>
                                            catalogTable,
                                            DynamoDbTable<MovieActor>
                                            movieActorTable) {

    // 1. Request three items from two tables using a builder.
    final List<Document> documents = enhancedClient.transactGetItems(b -> b
        .add.GetItem(catalogTable,
        Key.builder().partitionValue(2).sortValue("Title 55").build())
        .add.GetItem(movieActorTable, Key.builder().partitionValue("Sophie's
        Choice").sortValue("Meryl Streep").build())
        .add.GetItem(movieActorTable, Key.builder().partitionValue("Blue
        Jasmine").sortValue("Cate Blanchett").build())
        .build());
}
```

```
// 2. A list of Document objects is returned in the same order as requested.  
ProductCatalog title55 = documents.get(0).getItem(catalogTable);  
if (title55 != null) {  
    logger.info(title55.toString());  
}  
  
MovieActor sophiesChoice = documents.get(1).getItem(movieActorTable);  
if (sophiesChoice != null) {  
    logger.info(sophiesChoice.toString());  
}  
  
// 3. The getItem() method returns null if the Document object contains no item  
from DynamoDB.  
MovieActor blueJasmine = documents.get(2).getItem(movieActorTable);  
if (blueJasmine != null) {  
    logger.info(blueJasmine.toString());  
}  
}
```

As tabelas do DynamoDB contêm os seguintes itens antes da execução do exemplo de código.

```
ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}  
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best  
Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

A saída a seguir é registrada. Se um item for solicitado, mas não encontrado, ele não será devolvido, como é o caso da solicitação do filme nomeado `Blue Jasmine`.

```
ProductCatalog{id=2, title='Title 55', isbn='orig_isbn', authors=[b, g], price=10}  
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best  
Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

Exemplos do `transactWriteItems()`

O [`transactWriteItems\(\)`](#) aceita até 100 ações de colocar, atualizar ou excluir em uma única transação atômica em várias tabelas. O Amazon DynamoDB Developer Guide contém detalhes sobre restrições e condições de falha da operação do serviço [subjacente do DynamoDB](#).

Exemplo básico

No exemplo a seguir, quatro operações são solicitadas para duas tabelas. As classes de modelo correspondentes [`ProductCatalog`](#) [`MovieActor`](#) foram mostradas anteriormente.

Cada uma das três operações possíveis — colocar, atualizar e excluir — usa um parâmetro de solicitação dedicado para especificar os detalhes.

O código após a linha de comentário 1 mostra a variação simples do `addPutItem()` método. O método aceita um [MappedTableResource](#) objeto e a instância do objeto de dados para colocar. A declaração após a linha de comentário 2 mostra a variação que aceita uma [TransactPutItemEnhancedRequest](#) instância. Essa variação permite adicionar mais opções na solicitação, como uma expressão de condição. Um [exemplo](#) subsequente mostra uma expressão de condição para uma operação individual.

Uma operação de atualização é solicitada após a linha de comentários 3.

[TransactUpdateItemEnhancedRequest](#) tem um `ignoreNulls()` método que permite configurar o que o SDK faz com null os valores no objeto do modelo. Se o `ignoreNulls()` método retornar verdadeiro, o SDK não removerá os valores dos atributos da tabela para os atributos do objeto de dados que são null. Se o `ignoreNulls()` método retornar false, o SDK solicitará que o serviço do DynamoDB remova os atributos do item na tabela. O valor padrão para `ignoreNulls` é false.

A declaração após a linha de comentário 4 mostra a variação de uma solicitação de exclusão que usa um objeto de dados. O cliente aprimorado extrai os valores-chave antes de enviar a solicitação final.

```
public static void transactWriteItems(DynamoDbEnhancedClient enhancedClient,
                                      DynamoDbTable<ProductCatalog> catalogTable,
                                      DynamoDbTable<MovieActor> movieActorTable) {

    enhancedClient.transactWriteItems(b -> b
        // 1. Simplest variation of put item request.
        .addPutItem(catalogTable, getProductId2())
        // 2. Put item request variation that accommodates condition
        expressions.
        .addPutItem(movieActorTable,
        TransactPutItemEnhancedRequest.builder(MovieActor.class)
            .item(getMovieActorStreep())

        .conditionExpression(Expression.builder().expression("attribute_not_exists
        (movie)").build())
            .build())
        // 3. Update request that does not remove attribute values on the table
        if the data object's value is null.
```

```
.addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
    .item(getProductCatId4ForUpdate())
    .ignoreNulls(Boolean.TRUE)
    .build())
// 4. Variation of delete request that accepts a data object. The key
values are extracted for the request.
.addDeleteItem(movieActorTable, getMovieActorBlanchett())
);
}
```

Os métodos auxiliares a seguir fornecem os objetos de dados para os add*Item parâmetros.

Métodos auxiliares

```
public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Tar");
    movieActor.setActingYear(2022);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}

public static MovieActor getMovieActorStreep() {
```

```
        MovieActor movieActor = new MovieActor();
        movieActor.setActorName("Meryl Streep");
        movieActor.setMovieName("Sophie's Choice");
        movieActor.setActingYear(1982);
        movieActor.setActingAward("Best Actress");
        movieActor.setActingSchoolName("Yale School of Drama");
        return movieActor;
    }
```

As tabelas do DynamoDB contêm os seguintes itens antes da execução do exemplo de código.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

Os itens a seguir estão nas tabelas após a conclusão da execução do código.

```
3 | ProductCatalog{id=2, title='Title 55', isbn='1-565-85698', authors=[a, b],
price=30.22}
4 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=40.0}
5 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
```

O item na linha 2 foi excluído e as linhas 3 e 5 mostram os itens que foram colocados. A linha 4 mostra a atualização da linha 1. O price valor é o único valor que foi alterado no item. Se ignoreNulls() tivesse retornado false, a linha 4 ficaria parecida com a seguinte linha.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

Exemplo de verificação de condição

O exemplo a seguir mostra o uso de uma verificação de condição. Uma verificação de condição é usada para verificar se um item existe ou para verificar a condição de atributos específicos de um item no banco de dados. O item verificado na verificação de condição não pode ser usado em outra operação na transação.

Note

Você não pode visar o mesmo item com várias operações dentro da mesma transação. Por exemplo, você não pode realizar uma verificação de condição e também tentar atualizar o mesmo item na mesma transação.

O exemplo mostra um de cada tipo de operação em uma solicitação transacional de itens de gravação. Depois da linha de comentário 2, o `addConditionCheck()` método fornece a condição que falha na transação se o `conditionExpression` parâmetro for avaliado como `false`. A expressão de condição que é retornada do método mostrado no bloco de métodos auxiliares verifica se o ano de premiação do filme `Sophie's Choice` é igual a `1982`. Se for, a expressão será avaliada `false` e a transação falhará.

Este guia discute [expressões](#) em profundidade em outro tópico.

```
public static void conditionCheckFailExample(DynamoDbEnhancedClient enhancedClient,
                                              DynamoDbTable<ProductCatalog>
                                              catalogTable,
                                              DynamoDbTable<MovieActor>
                                              movieActorTable) {

    try {
        enhancedClient.transactWriteItems(b -> b
            // 1. Perform one of each type of operation with the next three
            methods.
            .addPutItem(catalogTable,
            TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
                .item(getProductCatId2()).build())
            .addUpdateItem(catalogTable,
            TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
                .item(getProductCatId4ForUpdate())
                .ignoreNulls(Boolean.TRUE).build())
            .addDeleteItem(movieActorTable,
            TransactDeleteItemEnhancedRequest.builder()
                .key(b1 -> b1

            .partitionValue(getMovieActorBlanchett().getMovieName())

            .sortValue(getMovieActorBlanchett().getActorName())).build())
    }
}
```

```
// 2. Add a condition check on a table item that is not involved in
another operation in this request.
    .addConditionCheck(movieActorTable, ConditionCheck.builder()
        .conditionExpression(buildConditionCheckExpression())
        .key(k -> k
            .partitionValue("Sophie's Choice")
            .sortValue("Meryl Streep"))
    // 3. Specify the request to return existing values from
the item if the condition evaluates to true.

.returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
    .build())
    .build());
// 4. Catch the exception if the transaction fails and log the information.
} catch (TransactionCanceledException ex) {
    ex.cancellationReasons().stream().forEach(cancellationReason -> {
        logger.info(cancellationReason.toString());
    });
}
}
```

Os métodos auxiliares a seguir são usados no exemplo de código anterior.

Métodos auxiliares

```
private static Expression buildConditionCheckExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(1982));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}
```

```
public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)
        .price(BigDecimal.valueOf(40.00))
        .title("Title 1")
        .build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2013);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}
```

As tabelas do DynamoDB contêm os seguintes itens antes da execução do exemplo de código.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
   Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
3 | MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
   actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

Os itens a seguir estão nas tabelas após a conclusão da execução do código.

```
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
MovieActor{movieName='Sophie's Choice', actorName='Meryl Streep', actingAward='Best
   Actress', actingYear=1982, actingSchoolName='Yale School of Drama'}
MovieActor{movieName='Tar', actorName='Cate Blanchett', actingAward='Best Actress',
   actingYear=2022, actingSchoolName='National Institute of Dramatic Art'}
```

Os itens permanecem inalterados nas tabelas porque a transação falhou. O `actingYear` valor do filme `Sophie's Choice` é 1982, conforme mostrado na linha 2 dos itens na tabela antes da chamada do `transactWriteItem()` método.

Para capturar as informações de cancelamento da transação, inclua a chamada do `transactWriteItems()` método em um `try` bloco e `catch` o.

[TransactionCanceledException](#) Depois da linha de comentário 4 do exemplo, o código registra

cada [CancellationReason](#) objeto. Como o código após a linha de comentário 3 do exemplo especifica que os valores devem ser retornados para o item que causou a falha da transação, o registro exibe os valores brutos do banco de dados do item do Sophie's Choice filme.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Meryl Streep),
    movie=AttributeValue(S=Sophie's Choice), actingaward=AttributeValue(S=Best Actress),
    actingyear=AttributeValue(N=1982), actingschoolname=AttributeValue(S=Yale School of
    Drama)}, -
    Code=ConditionalCheckFailed, Message=The conditional request failed.)
```

Exemplo de condição de operação única

O exemplo a seguir mostra o uso de uma condição em uma única operação em uma solicitação de transação. A operação de exclusão após a linha de comentário 1 contém uma condição que verifica o valor do item de destino da operação em relação ao banco de dados. Neste exemplo, a expressão de condição criada com o método auxiliar após a linha de comentário 2 especifica que o item deve ser excluído do banco de dados se o ano de atuação do filme não for igual a 2013.

As [expressões](#) serão discutidas posteriormente neste guia.

```
public static void singleOperationConditionFailExample(DynamoDbEnhancedClient
enhancedClient,
DynamoDbTable<ProductCatalog> catalogTable,
DynamoDbTable<MovieActor>
movieActorTable) {
    try {
        enhancedClient.transactWriteItems(b -> b
            .addPutItem(catalogTable,
TransactPutItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId2())
            .build())
            .addUpdateItem(catalogTable,
TransactUpdateItemEnhancedRequest.builder(ProductCatalog.class)
            .item(getProductCatId4ForUpdate())
            .ignoreNulls(Boolean.TRUE).build())
        // 1. Delete operation that contains a condition expression
            .addDeleteItem(movieActorTable,
TransactDeleteItemEnhancedRequest.builder()
```

```

        .key((Key.Builder k) -> {
            MovieActor blanchett = getMovieActorBlanchett();
            k.partitionValue(blanchett.getMovieName())
                .sortValue(blanchett.getActorName());
        })
        .conditionExpression(buildDeleteItemExpression())

.returnValuesOnConditionCheckFailure(ReturnValuesOnConditionCheckFailure.ALL_OLD)
        .build()
    .build();
} catch (TransactionCanceledException ex) {
    ex.cancellationReasons().forEach(cancellationReason ->
logger.info(cancellationReason.toString()));
}
}

// 2. Provide condition expression to check if 'actingyear' is not equal to 2013.
private static Expression buildDeleteItemExpression() {
    Map<String, AttributeValue> expressionValue = Map.of(
        ":year", numberValue(2013));

    return Expression.builder()
        .expression("actingyear <> :year")
        .expressionValues(expressionValue)
        .build();
}
}

```

Os métodos auxiliares a seguir são usados no exemplo de código anterior.

Métodos auxiliares

```

public static ProductCatalog getProductCatId2() {
    return ProductCatalog.builder()
        .id(2)
        .isbn("1-565-85698")
        .authors(new HashSet<>(Arrays.asList("a", "b")))
        .price(BigDecimal.valueOf(30.22))
        .title("Title 55")
        .build();
}

public static ProductCatalog getProductCatId4ForUpdate() {
    return ProductCatalog.builder()
        .id(4)

```

```
.price(BigDecimal.valueOf(40.00))
.title("Title 1")
.build();
}

public static MovieActor getMovieActorBlanchett() {
    MovieActor movieActor = new MovieActor();
    movieActor.setActorName("Cate Blanchett");
    movieActor.setMovieName("Blue Jasmine");
    movieActor.setActingYear(2013);
    movieActor.setActingAward("Best Actress");
    movieActor.setActingSchoolName("National Institute of Dramatic Art");
    return movieActor;
}
```

As tabelas do DynamoDB contêm os seguintes itens antes da execução do exemplo de código.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
```

Os itens a seguir estão nas tabelas após a conclusão da execução do código.

```
ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2023-03-15 11:29:07 [main] INFO org.example.tests.TransactDemoTest:168 -
MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
```

Os itens permanecem inalterados nas tabelas porque a transação falhou. O `actingYear` valor do filme `Blue Jasmine` é `2013` mostrado na linha 2 da lista de itens antes da execução do exemplo de código.

As linhas a seguir são registradas no console.

```
CancellationReason(Code=None)
CancellationReason(Code=None)
CancellationReason(Item={actor=AttributeValue(S=Cate Blanchett),
movie=AttributeValue(S=Blue Jasmine), actingaward=AttributeValue(S=Best Actress),
actingyear=AttributeValue(N=2013), actingschoolname=AttributeValue(S=National
Institute of Dramatic Art)},
Code=ConditionalCheckFailed, Message=The conditional request failed)
```

Índices secundários

Os índices secundários melhoram o acesso aos dados definindo chaves alternativas que você usa nas operações de consulta e varredura. Os índices secundários globais (GSI) têm uma chave de partição e uma chave de classificação que podem ser diferentes das da tabela base. Por outro lado, os índices secundários locais (LSI) usam a chave de partição do índice primário.

Anote a classe de dados com anotações de índice secundário

Os atributos que participam de índices secundários exigem a `@DynamoDbSecondarySortKey` anotação `@DynamoDbSecondaryPartitionKey` ou.

A classe a seguir mostra anotações para dois índices. O GSI nomeado `SubjectLastPostedDateIndex` usa o `Subject` atributo para a chave de partição e o `LastPostedDateTime` para a chave de classificação. O LSI nomeado `ForumLastPostedDateIndex` usa o `ForumName` como chave de partição e `LastPostedDateTime` como chave de classificação.

Observe que o `Subject` atributo tem uma função dupla. É a chave de classificação da chave primária e a chave de partição do GSI nomeado `SubjectLastPostedDateIndex`.

Classe MessageThread

A `MessageThread` classe é adequada para ser usada como uma classe de dados para a [tabela Thread de exemplo](#) no Amazon DynamoDB Developer Guide.

Importações

```
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.DynamoDbSecondaryPartitionKey;
import
software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSecondarySortKey;
import software.amazon.awssdk.enhanced.dynamodb.mapper.annotations.DynamoDbSortKey;

import java.util.List;
```

```
@DynamoDbBean
public class MessageThread {
    private String ForumName;
```

```
private String Subject;
private String Message;
private String LastPostedBy;
private String LastPostedDateTime;
private Integer Views;
private Integer Replies;
private Integer Answered;
private List<String> Tags;

@DynamoDbPartitionKey
public String getForumName() {
    return ForumName;
}

public void setForumName(String forumName) {
    ForumName = forumName;
}

// Sort key for primary index and partition key for GSI
"SubjectLastPostedDateIndex".

@DynamoDbSortKey
@DynamoDbSecondaryPartitionKey(indexNames = "SubjectLastPostedDateIndex")
public String getSubject() {
    return Subject;
}

public void setSubject(String subject) {
    Subject = subject;
}

// Sort key for GSI "SubjectLastPostedDateIndex" and sort key for LSI
"ForumLastPostedDateIndex".
@DynamoDbSecondarySortKey(indexNames = {"SubjectLastPostedDateIndex",
"ForumLastPostedDateIndex"})
public String getLastPostedDateTime() {
    return LastPostedDateTime;
}

public void setLastPostedDateTime(String lastPostedDateTime) {
    LastPostedDateTime = lastPostedDateTime;
}

public String getMessage() {
    return Message;
}
```

```
public void setMessage(String message) {
    Message = message;
}

public String getLastPostedBy() {
    return LastPostedBy;
}

public void setLastPostedBy(String lastPostedBy) {
    LastPostedBy = lastPostedBy;
}

public Integer getViews() {
    return Views;
}

public void setViews(Integer views) {
    Views = views;
}

@DynamoDbSecondaryPartitionKey(indexNames = "ForumRepliesIndex")
public Integer getReplies() {
    return Replies;
}

public void setReplies(Integer replies) {
    Replies = replies;
}

public Integer getAnswered() {
    return Answered;
}

public void setAnswered(Integer answered) {
    Answered = answered;
}

public List<String> getTags() {
    return Tags;
}

public void setTags(List<String> tags) {
    Tags = tags;
```

```
}

public MessageThread() {
    this.Answered = 0;
    this.LastPostedBy = "";
    this.ForumName = "";
    this.Message = "";
    this.LastPostedDateTime = "";
    this.Replies = 0;
    this.Views = 0;
    this.Subject = "";
}

@Override
public String toString() {
    return "MessageThread{" +
        "ForumName='" + ForumName + '\'' +
        ", Subject='" + Subject + '\'' +
        ", Message='" + Message + '\'' +
        ", LastPostedBy='" + LastPostedBy + '\'' +
        ", LastPostedDateTime='" + LastPostedDateTime + '\'' +
        ", Views=" + Views +
        ", Replies=" + Replies +
        ", Answered=" + Answered +
        ", Tags=" + Tags +
        '}';
}
}
```

Crie o índice

A partir da versão 2.20.86 do SDK for Java, o `createTable()` método gera automaticamente índices secundários a partir de anotações de classes de dados. Por padrão, todos os atributos da tabela base são copiados para um índice e os valores de taxa de transferência provisionados são 20 unidades de capacidade de leitura e 20 unidades de capacidade de gravação.

No entanto, se você usar uma versão do SDK anterior à 2.20.86, precisará criar o índice junto com a tabela, conforme mostrado no exemplo a seguir. Este exemplo cria os dois índices para a `Thread` tabela. O parâmetro [builder](#) tem métodos para configurar os dois tipos de índices, conforme mostrado após as linhas de comentário 1 e 2. Você usa o `indexName()` método do criador de índices para associar os nomes de índice especificados nas anotações da classe de dados ao tipo de índice pretendido.

Esse código configura todos os atributos da tabela para terminarem nos dois índices após as linhas de comentário 3 e 4. Mais informações sobre [projeções de atributos](#) estão disponíveis no Amazon DynamoDB Developer Guide.

```
public static void createMessageThreadTable(DynamoDbTable<MessageThread>
messageThreadDynamoDbTable, DynamoDbClient dynamoDbClient) {
    messageThreadDynamoDbTable.createTable(b -> b
        // 1. Generate the GSI.
        .globalSecondaryIndices(gsi ->
            gsi.indexName("SubjectLastPostedDateIndex")
                // 3. Populate the GSI with all attributes.
                .projection(p -> p
                    .projectionType(ProjectionType.ALL))
        )
        // 2. Generate the LSI.
        .localSecondaryIndices(lsi -> lsi.indexName("ForumLastPostedDateIndex"))
            // 4. Populate the LSI with all attributes.
            .projection(p -> p
                .projectionType(ProjectionType.ALL))
    )
};

});
```

Consulte usando um índice

O exemplo a seguir consulta o índice ForumLastPostedDateIndexsecundário local.

Seguindo a linha de comentário 2, você cria um [QueryConditional](#) objeto que é necessário ao chamar o método [DynamoDbIndex.query\(\)](#).

Você obtém uma referência ao índice que deseja consultar após a linha de comentário 3 ao passar o nome do índice. Seguindo a linha de comentário 4, você chama o query() método no índice que passa o QueryConditional objeto.

Você também configura a consulta para retornar três valores de atributos, conforme mostrado após a linha de comentário 5. Se não attributesToProject() for chamada, a consulta retornará todos os valores dos atributos. Observe que os nomes dos atributos especificados começam com letras minúsculas. Esses nomes de atributos correspondem aos usados na tabela, não necessariamente aos nomes dos atributos da classe de dados.

Seguindo a linha de comentário 6, repita os resultados e registre cada item retornado pela consulta e também o armazene na lista para retornar ao chamador.

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient  
enhancedClient,  
                                         String lastPostedDate,  
  
DynamoDbTable<MessageThread> threadTable) {  
    // 1. Log the parameter value.  
    logger.info("lastPostedDate value: {}", lastPostedDate);  
  
    // 2. Create a QueryConditional whose sort key value must be greater than or  
    // equal to the parameter value.  
    QueryConditional queryConditional =  
QueryConditional.sortGreaterThanOrEqualTo(qc ->  
    qc.partitionValue("Forum02").sortValue(lastPostedDate));  
  
    // 3. Specify the index name to query the DynamoDbIndex instance.  
    final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =  
threadTable.index("ForumLastPostedDateIndex");  
  
    // 4. Perform the query by using the QueryConditional object.  
    finalSdkIterable<Page<MessageThread>> pagedResult =  
forumLastPostedDateIndex.query(q -> q  
    .queryConditional(queryConditional)  
    // 5. Request three attribute in the results.  
    .attributesToProject("forumName", "subject", "lastPostedDateTime"));  
  
    List<MessageThread> collectedItems = new ArrayList<>();  
    // 6. Iterate through the pages response and sort the items.  
    pagedResult.stream().forEach(page -> page.items().stream()  
  
.sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))  
    .forEach(mt -> {  
        // 7. Log the returned items and add the collection to  
        // return to the caller.  
        logger.info(mt.toString());  
        collectedItems.add(mt);  
    });  
    return collectedItems;  
}
```

Os itens a seguir existem no banco de dados antes da execução da consulta.

```
MessageThread{ForumName='Forum01', Subject='Subject01', Message='Message01',
LastPostedBy='', LastPostedDateTime='2023.03.28', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject02', Message='Message02',
LastPostedBy='', LastPostedDateTime='2023.03.29', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject04', Message='Message04',
LastPostedBy='', LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='Message08',
LastPostedBy='', LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='Message10',
LastPostedBy='', LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject03', Message='Message03',
LastPostedBy='', LastPostedDateTime='2023.03.30', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject06', Message='Message06',
LastPostedBy='', LastPostedDateTime='2023.04.02', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum03', Subject='Subject09', Message='Message09',
LastPostedBy='', LastPostedDateTime='2023.04.05', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum05', Subject='Subject05', Message='Message05',
LastPostedBy='', LastPostedDateTime='2023.04.01', Views=0, Replies=0, Answered=0,
Tags=null}
MessageThread{ForumName='Forum07', Subject='Subject07', Message='Message07',
LastPostedBy='', LastPostedDateTime='2023.04.03', Views=0, Replies=0, Answered=0,
Tags=null}
```

As instruções de registro nas linhas 1 e 6 resultam na seguinte saída do console.

```
lastPostedDate value: 2023.03.31
MessageThread{ForumName='Forum02', Subject='Subject04', Message='', LastPostedBy='',
LastPostedDateTime='2023.03.31', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject08', Message='', LastPostedBy='',
LastPostedDateTime='2023.04.04', Views=0, Replies=0, Answered=0, Tags=null}
MessageThread{ForumName='Forum02', Subject='Subject10', Message='', LastPostedBy='',
LastPostedDateTime='2023.04.06', Views=0, Replies=0, Answered=0, Tags=null}
```

A consulta retornou itens com um *forumName* valor de Forum02 e um *LastPostedDateTime* valor maior ou igual a 2023.03.31. Os resultados mostram message valores com uma string vazia, embora os message atributos tenham valores no índice. Isso ocorre porque o atributo da mensagem não foi projetado após a linha de comentário 5.

Extensões

A API do DynamoDB Enhanced Client suporta extensões de plug-in que fornecem funcionalidades além das operações de mapeamento. As extensões têm dois métodos de gancho, `beforeWrite()` `afterRead()` e `beforeWrite()` modifica uma operação de gravação antes que ela aconteça, e o `afterRead()` método modifica os resultados de uma operação de leitura depois que ela acontece. Como algumas operações (como atualizações de itens) realizam uma gravação e depois uma leitura, os dois métodos de gancho são chamados.

As extensões são carregadas na ordem em que são especificadas no construtor de clientes aprimorado. A ordem de carregamento pode ser importante porque uma extensão pode atuar em valores que foram transformados por uma extensão anterior.

A API de cliente aprimorada vem com um conjunto de extensões de plug-in localizadas no [extensions](#) pacote. Por padrão, o cliente aprimorado carrega o [VersionedRecordExtension](#) e [AtomicCounterExtension](#) o. Você pode substituir o comportamento padrão com o construtor de clientes aprimorado e carregar qualquer extensão. Você também pode especificar nenhuma se não quiser as extensões padrão.

Se você carregar suas próprias extensões, o cliente aprimorado não carregará nenhuma extensão padrão. Se você quiser o comportamento fornecido por qualquer extensão padrão, precisará adicioná-la explicitamente à lista de extensões.

No exemplo a seguir, uma extensão personalizada chamada `verifyChecksumExtension` é carregada após a `VersionedRecordExtension`, que geralmente é carregada por padrão sozinha. O não `AtomicCounterExtension` está carregado neste exemplo.

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
                    verifyChecksumExtension)
```

```
.build();
```

VersionedRecordExtension

O VersionedRecordExtension é carregado por padrão e incrementará e rastreará o número da versão do item à medida que os itens forem gravados no banco de dados. Uma condição será adicionada a cada gravação que faz com que a gravação falhe se o número da versão do item persistente real não corresponder ao valor que o aplicativo leu pela última vez. Esse comportamento fornece efetivamente um bloqueio otimista para atualizações de itens. Se outro processo atualizar um item entre o momento em que o primeiro processo lê o item e está gravando uma atualização nele, a gravação falhará.

Para especificar qual atributo usar para rastrear o número da versão do item, marque um atributo numérico no esquema da tabela.

O trecho a seguir especifica que o `version` atributo deve conter o número da versão do item.

```
@DynamoDbVersionAttribute  
public Integer getVersion() {...};  
public void setVersion(Integer version) {...};
```

A abordagem equivalente do esquema de tabela estática é mostrada no trecho a seguir.

```
.addAttribute(Integer.class, a -> a.name("version")  
                .getter(Customer::getVersion)  
                .setter(Customer::setVersion)  
                // Apply the 'version' tag to the attribute.  
  
.tags(VersionedRecordExtension.AttributeTags.versionAttribute())
```

AtomicCounterExtension

O AtomicCounterExtension é carregado por padrão e incrementa um atributo numérico marcado sempre que um registro é gravado no banco de dados. Os valores iniciais e de incremento podem ser especificados. Se nenhum valor for especificado, o valor inicial será definido como 0 e o valor do atributo será incrementado em 1.

Para especificar qual atributo é um contador, marque um atributo do tipo Long no esquema da tabela.

O trecho a seguir mostra o uso dos valores padrão de início e incremento para o atributo `counter`

```
@DynamoDbAtomicCounter  
public Long getCounter() {...};  
public void setCounter(Long counter) {...};
```

A abordagem do esquema de tabela estática é mostrada no trecho a seguir. A extensão do contador atômico usa um valor inicial de 10 e incrementa o valor em 5 cada vez que o registro é gravado.

```
.addAttribute(Integer.class, a -> a.name("counter")  
                .getter(Customer::getCounter)  
                .setter(Customer::setCounter)  
                // Apply the 'atomicCounter' tag to the  
attribute with start and increment values.  
                .tags(StaticAttributeTags.atomicCounter(10L,  
5L))
```

AutoGeneratedTimestampRecordExtension

O atualiza `AutoGeneratedTimestampRecordExtension` automaticamente os atributos marcados do tipo [Instant](#) com um carimbo de data/hora atual sempre que o item é gravado com sucesso no banco de dados.

Essa extensão não é carregada por padrão. Portanto, você precisa especificá-la como uma extensão personalizada ao criar o cliente aprimorado, conforme mostrado no primeiro exemplo deste tópico.

Para especificar qual atributo atualizar com o timestamp atual, marque o `Instant` atributo no esquema da tabela.

O `lastUpdate` atributo é o alvo do comportamento das extensões no trecho a seguir. Observe a exigência de que o atributo seja um `Instant` tipo.

```
@DynamoDbAutoGeneratedTimestampAttribute  
public Instant getLastUpdate() {...}  
public void setLastUpdate(Instant lastUpdate) {...}
```

A abordagem equivalente do esquema de tabela estática é mostrada no trecho a seguir.

```
.addAttribute(Instant.class, a -> a.name("lastUpdate")  
                .getter(Customer::getLastUpdate)
```

```
        .setter(Customer::setLastUpdate)
        // Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

Extensões personalizadas

A classe de extensão personalizada a seguir mostra um `beforeWrite()` método que usa uma expressão de atualização. Após a linha de comentário 2, criamos um `SetAction` para definir o `registrationDate` atributo se o item no banco de dados ainda não tiver um `registrationDate` atributo. Sempre que um `Customer` objeto é atualizado, a extensão garante que a `registrationDate` esteja definido.

```
public final class CustomExtension implements DynamoDbEnhancedClientExtension {

    // 1. In a custom extension, use an UpdateExpression to define what action to take
    // before
    //     an item is updated.
    @Override
    public WriteModification beforeWrite(DynamoDbExtensionContext.BeforeWrite context)
    {
        if ( context.operationContext().tableName().equals("Customer")
            && context.operationName().equals(OperationName.UPDATE_ITEM) ) {
            return WriteModification.builder()
                .updateExpression(createUpdateExpression())
                .build();
        }
        return WriteModification.builder().build(); // Return an "empty"
        // WriteModification instance if the extension should not be applied.
                                                // In this case, if the code is
not updating an item on the Customer table.
    }

    private static UpdateExpression createUpdateExpression() {

        // 2. Use a SetAction, a subclass of UpdateAction, to provide the values in the
        // update.
        SetAction setAction =
            SetAction.builder()
                .path("registrationDate")
                .value("if_not_exists(registrationDate, :regValue)")
```

```
        .putExpressionValue(":regValue",
AttributeValue.fromS(Instant.now().toString()))
        .build();
    // 3. Build the UpdateExpression with one or more UpdateAction.
    return UpdateExpression.builder()
        .addAction(setAction)
        .build();
}
}
```

Recursos avançados de esquema de tabela

Saiba mais sobre os recursos avançados do esquema de tabelas na API do DynamoDB Enhanced Client.

Visão geral do esquema

[TableSchema](#) é a interface para a funcionalidade de mapeamento da API do DynamoDB Enhanced Client. Ele pode mapear um objeto de dados de e para um mapa de [AttributeValues](#). Um TableSchema objeto precisa conhecer a estrutura da tabela que está mapeando. Essas informações de estrutura são armazenadas em um [TableMetadata](#) objeto.

A API de cliente aprimorada tem várias implementações de TableSchema, que seguem.

Esquema de tabela gerado a partir de classes anotadas

É uma operação moderadamente cara criar uma a TableSchema partir de classes anotadas, portanto, recomendamos fazer isso uma vez, na inicialização do aplicativo.

[BeanTableSchema](#)

Essa implementação é construída com base nos atributos e anotações de uma classe de bean. Um exemplo dessa abordagem é demonstrado na [seção Introdução](#).

Note

Se a não BeanTableSchema estiver se comportando conforme o esperado, habilite o registro de depuração para.
`software.amazon.awssdk.enhanced.dynamodb.beans`

[ImmutableTableSchema](#)

Essa implementação é criada a partir de uma classe de dados imutável. Essa abordagem é descrita na [???](#) seção.

Esquema de tabela gerado com um construtor

Os seguintes TableSchema s são criados a partir do código usando um construtor. Essa abordagem é menos dispendiosa do que a abordagem que usa classes de dados anotadas. A abordagem do construtor evita o uso de anotações e não exige JavaBean padrões de nomenclatura.

[StaticTableSchema](#)

Essa implementação foi criada para classes de dados mutáveis. A seção de introdução deste guia demonstrou como [gerar um StaticTableSchema usando um construtor](#).

[StaticImmutableTableSchema](#)

Da mesma forma que você cria umStaticTableSchema, você gera uma implementação desse tipo de uso de TableSchema um [construtor](#) para uso com classes de dados imutáveis.

Esquema de tabela para dados sem um esquema fixo

[DocumentTableSchema](#)

Ao contrário de outras implementações doTableSchema, você não define atributos para uma DocumentTableSchema instância. Normalmente, você especifica somente chaves primárias e provedores de conversão de atributos. Uma EnhancedDocument instância fornece os atributos que você cria a partir de elementos individuais ou de uma string JSON.

Inclua ou exclua atributos explicitamente

A API do DynamoDB Enhanced Client oferece anotações para impedir que atributos da classe de dados se tornem atributos em uma tabela. Com a API, você também pode usar um nome de atributo diferente do nome do atributo da classe de dados.

Excluir atributos

Para ignorar atributos que não devem ser mapeados para uma tabela do DynamoDB, marque o atributo com a anotação. `@DynamoDbIgnore`

```
private String internalKey;  
  
@DynamoDbIgnore  
public String getInternalKey() { return this.internalKey; }  
public void setInternalKey(String internalKey) { return this.internalKey =  
    internalKey; }
```

Incluir atributos

Para alterar o nome de um atributo usado na tabela do DynamoDB, marque-o com a anotação e `@DynamoDbAttribute` forneça um nome diferente.

```
private String internalKey;  
  
@DynamoDbAttribute("renamedInternalKey")  
public String getInternalKey() { return this.internalKey; }  
public void setInternalKey(String internalKey) { return this.internalKey =  
    internalKey; }
```

Conversão de atributos de controle

Por padrão, um esquema de tabela fornece conversores para todos os tipos primitivos e muitos tipos comuns de Java por meio de uma implementação padrão da interface.

[AttributeConverterProvider](#) Você pode alterar o comportamento padrão geral com uma `AttributeConverterProvider` implementação personalizada. Você também pode alterar o conversor para um único atributo.

Para obter uma lista dos conversores disponíveis, consulte a [AttributeConverter](#)interface Java doc.

Forneça provedores de conversão de atributos personalizados

Você pode fornecer um único `AttributeConverterProvider` ou uma cadeia de `AttributeConverterProvider`s ordenados por meio da `@DynamoDbBean` (`converterProviders = {...}`) anotação. Qualquer personalização `AttributeConverterProvider` deve estender a `AttributeConverterProvider` interface.

Observe que, se você fornecer sua própria cadeia de provedores de conversão de atributos, substituirá o provedor de conversão padrão, `DefaultAttributeConverterProvider`. Se quiser usar a funcionalidade do `DefaultAttributeConverterProvider`, você deve incluí-la na cadeia.

Também é possível anotar o bean com uma matriz vazia. {} Isso desativa o uso de qualquer provedor de conversão de atributos, incluindo o padrão. Nesse caso, todos os atributos a serem mapeados devem ter seu próprio conversor de atributos.

O trecho a seguir mostra um único provedor de conversor.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

O trecho a seguir mostra o uso de uma cadeia de provedores de conversores. Como o SDK padrão é fornecido por último, ele tem a prioridade mais baixa.

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {

}
```

Os criadores de esquemas de tabelas estáticas têm um `attributeConverterProviders()` método que funciona da mesma maneira. Isso é mostrado no trecho a seguir.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =
    StaticTableSchema.builder(Customer.class)
        .newItemSupplier(Customer::new)
        .addAttribute(String.class, a -> a.name("name")
            a.getter(Customer::getName)
            a.setter(Customer::setName))
        .attributeConverterProviders(converterProvider1, converterProvider2)
        .build();
```

Substituir o mapeamento de um único atributo

Para substituir a forma como um único atributo é mapeado, forneça um `AttributeConverter` para o atributo. Essa adição substitui todos os conversores fornecidos pelo esquema `AttributeConverterProviders` da tabela. Isso adiciona um conversor personalizado somente para esse atributo. Outros atributos, mesmo aqueles do mesmo tipo, não usarão esse conversor, a menos que ele seja explicitamente especificado para esses outros atributos.

A `@DynamoDbConvertedBy` anotação é usada para especificar a `AttributeConverter` classe personalizada, conforme mostrado no trecho a seguir.

```
@DynamoDbBean  
public class Customer {  
    private String name;  
  
    @DynamoDbConvertedBy(CustomAttributeConverter.class)  
    public String getName() { return this.name; }  
    public void setName(String name) { this.name = name; }  
}
```

Os construtores de esquemas estáticos têm um método construtor `attributeConverter()` de atributos equivalente. Esse método usa uma instância de `umAttributeConverter`, conforme mostrado a seguir.

```
private static final StaticTableSchema<Customer> CUSTOMER_TABLE_SCHEMA =  
    StaticTableSchema.builder(Customer.class)  
        .newItemSupplier(Customer::new)  
        .addAttribute(String.class, a -> a.name("name")  
            a.getter(Customer::getName)  
            a.setter(Customer::setName)  
            a.attributeConverter(customAttributeConverter))  
        .build();
```

Exemplo

Este exemplo mostra uma `AttributeConverterProvider` implementação que fornece um conversor de atributos para `java.net.HttpCookie` objetos.

A `SimpleUser` classe a seguir contém um atributo chamado `lastUsedCookie` que é uma instância de `HttpCookie`.

O parâmetro para as `@DynamoDbBean` anotações lista as duas `AttributeConverterProvider` classes que fornecem conversores.

Class with annotations

```
@DynamoDbBean(converterProviders = {CookieConverterProvider.class,  
DefaultAttributeConverterProvider.class})
```

```
public static final class SimpleUser {  
    private String name;  
    private HttpCookie lastUsedCookie;  
  
    @DynamoDbPartitionKey  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public HttpCookie getLastUsedCookie() {  
        return lastUsedCookie;  
    }  
  
    public void setLastUsedCookie(HttpCookie lastUsedCookie) {  
        this.lastUsedCookie = lastUsedCookie;  
    }  
}
```

Static table schema

```
private static final TableSchema<SimpleUser> SIMPLE_USER_TABLE_SCHEMA =  
    TableSchema.builder(SimpleUser.class)  
        .newItemSupplier(SimpleUser::new)  
        .attributeConverterProviders(CookieConverterProvider.create(),  
            AttributeConverterProvider.defaultProvider())  
        .addAttribute(String.class, a -> a.name("name")  
            .setter(SimpleUser::setName)  
            .getter(SimpleUser::getName)  
            .tags(StaticAttributeTags.primaryPartitionKey()))  
        .addAttribute(HttpCookie.class, a -> a.name("lastUsedCookie")  
            .setter(SimpleUser::setLastUsedCookie)  
            .getter(SimpleUser::getLastUsedCookie))  
        .build();
```

O `CookieConverterProvider` exemplo a seguir fornece uma instância de `umHttpCookieConverter`.

```
public static final class CookieConverterProvider implements  
AttributeConverterProvider {
```

```
private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =  
ImmutableMap.of(  
    // 1. Add HttpCookieConverter to the internal cache.  
    EnhancedType.of(HttpCookie.class), new HttpCookieConverter());  
  
public static CookieConverterProvider create() {  
    return new CookieConverterProvider();  
}  
  
// The SDK calls this method to find out if the provider contains a  
AttributeConverter instance  
// for the EnhancedType<T> argument.  
@SuppressWarnings("unchecked")  
@Override  
public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {  
    return (AttributeConverter<T>) converterCache.get(enhancedType);  
}  
}
```

Código de conversão

No `transformFrom()` método da `HttpCookieConverter` classe a seguir, o código recebe uma `HttpCookie` instância e a transforma em um mapa do DynamoDB que é armazenado como um atributo.

O `transformTo()` método recebe um parâmetro de mapa do DynamoDB e, em seguida, invoca `HttpCookie` o construtor que exige um nome e um valor.

```
public static final class HttpCookieConverter implements  
AttributeConverter<HttpCookie> {  
  
    @Override  
    public AttributeValue transformFrom(HttpCookie httpCookie) {  
  
        return AttributeValue.fromM(  
            Map.of("cookieName", AttributeValue.fromS(httpCookie.getName()),  
                  "cookieValue", AttributeValue.fromS(httpCookie.getValue()))  
        );  
    }  
  
    @Override  
    public HttpCookie transformTo(AttributeValue attributeValue) {  
        Map<String, AttributeValue> map = attributeValue.m();  
    }  
}
```

```
        return new HttpCookie(
            map.get("cookieName").s(),
            map.get("cookieValue").s());
    }

    @Override
    public EnhancedType<HttpCookie> type() {
        return EnhancedType.of(HttpCookie.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.M;
    }
}
```

Alterar o comportamento de atualização dos atributos

Você pode personalizar o comportamento de atualização de atributos individuais ao realizar uma operação de atualização. [Alguns exemplos de operações de atualização na API de cliente aprimorada do DynamoDB são updateItem \(\) e \(\) .transactWriteItems](#)

Por exemplo, imagine que você queira armazenar um carimbo de data/hora criado em seu registro. No entanto, você deseja que seu valor seja gravado somente se não houver nenhum valor existente para o atributo já no banco de dados. Nesse caso, você usa o comportamento de [WRITE_IF_NOT_EXISTS](#) atualização.

O exemplo a seguir mostra a anotação que adiciona o comportamento ao `createdOn` atributo.

```
@DynamoDbBean
public class Customer extends GenericRecord {
    private String id;
    private Instant createdOn;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }
    public void setId(String id) { this.name = id; }

    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public Instant getCreatedOn() { return this.createdOn; }
    public void setCreatedOn(Instant createdOn) { this.createdOn = createdOn; }
}
```

Você pode declarar o mesmo comportamento de atualização ao criar um esquema de tabela estática, conforme mostrado no exemplo a seguir após a linha de comentário 1.

```
static final TableSchema<Customer> CUSTOMER_TABLE_SCHEMA =  
    TableSchema.builder(Customer.class)  
        .newItemSupplier(Customer::new)  
        .addAttribute(String.class, a -> a.name("id")  
            .getter(Customer::getId)  
            .setter(Customer::setId))  
  
.tags(StaticAttributeTags.primaryPartitionKey())  
    .addAttribute(Instant.class, a -> a.name("createdOn")  
        .getter(Customer::getCreatedOn)  
        .setter(Customer::setCreatedOn))  
    // 1. Add an UpdateBehavior.  
  
.tags(StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS))  
    .build();
```

Nivelar atributos de outras classes

Se os atributos da sua tabela estiverem espalhados por várias classes Java diferentes, seja por herança ou composição, a API do DynamoDB Enhanced Client fornece suporte para nivelar os atributos em uma classe.

Use herança

Se suas classes usam herança, use as seguintes abordagens para nivelar a hierarquia.

Use feijões anotados

Para a abordagem de anotação, ambas as classes devem conter a @DynamoDbBean anotação e uma classe deve conter uma ou mais anotações de chave primária.

Veja a seguir exemplos de classes de dados que têm uma relação de herança.

Standard data class

```
@DynamoDbBean  
public class Customer extends GenericRecord {  
    private String name;  
  
    public String getName() { return name; }
```

```
public void setName(String name) { this.name = name; }
}

@DynamoDbBean
public abstract class GenericRecord {
    private String id;
    private String createdDate;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
        createdDate; }
}
```

Lombok

A [onMethodopção](#) do Lombok copia anotações do DynamoDB baseadas em atributos, como, no código gerado. `@DynamoDbPartitionKey`

```
@DynamoDbBean
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord {
    private String name;
}

@Data
@DynamoDbBean
public abstract class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdDate;
}
```

Use esquemas estáticos

Para a abordagem do esquema estático, use o `extend()` método do construtor para reduzir os atributos da classe principal na classe secundária. Isso é mostrado após a linha de comentário 1 no exemplo a seguir.

```
StaticTableSchema<org.example.tests.model.inheritance.stat.GenericRecord>
GENERIC_RECORD_SCHEMA =  
  
StaticTableSchema.builder(org.example.tests.model.inheritance.stat.GenericRecord.class)
    // The partition key will be inherited by the top level mapper.
    .addAttribute(String.class, a -> a.name("id"))  
  
.getter(org.example.tests.model.inheritance.stat.GenericRecord::getId)  
  
.setter(org.example.tests.model.inheritance.stat.GenericRecord::setId)
    .tags(primaryPartitionKey())
    .addAttribute(String.class, a -> a.name("created_date"))  
  
.getter(org.example.tests.model.inheritance.stat.GenericRecord::getCreatedDate)  
  
.setter(org.example.tests.model.inheritance.stat.GenericRecord::setCreatedDate))
    .build();  
  
StaticTableSchema<org.example.tests.model.inheritance.stat.Customer>
CUSTOMER_SCHEMA =  
  
StaticTableSchema.builder(org.example.tests.model.inheritance.stat.Customer.class)
    .newItemSupplier(org.example.tests.model.inheritance.stat.Customer::new)
    .addAttribute(String.class, a -> a.name("name"))  
  
.getter(org.example.tests.model.inheritance.stat.Customer::getName)  
  
.setter(org.example.tests.model.inheritance.stat.Customer::setName))
    // 1. Use the extend() method to collapse the parent attributes
onto the child class.
    .extend(GENERIC_RECORD_SCHEMA)      // All the attributes of the
GenericRecord schema are added to Customer.
    .build();
```

O exemplo anterior do esquema estático usa as seguintes classes de dados. Como o mapeamento é definido quando você cria o esquema de tabela estática, as classes de dados não exigem anotações.

Classes de dados

Standard data class

```
public class Customer extends GenericRecord {
```

```
private String name;

public String getName() { return name; }
public void setName(String name) { this.name = name; }
}

public abstract class GenericRecord {
    private String id;
    private String createdDate;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
        createdDate; }
}
```

Lombok

```
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdDate;
}
```

Use a composição

Se suas classes usam composição, use as seguintes abordagens para nivelar a hierarquia.

Use feijões anotados

A `@DynamoDbFlatten` anotação nivela a classe contida.

Os exemplos de classes de dados a seguir usam a `@DynamoDbFlatten` anotação para adicionar efetivamente todos os atributos da `GenericRecord` classe contida à `Customer` classe.

Standard data class

```
@DynamoDbBean
public class Customer {
    private String name;
    private GenericRecord record;

    public String getName() { return this.name; }
    public void setName(String name) { this.name = name; }

    @DynamoDbFlatten
    public GenericRecord getRecord() { return this.record; }
    public void setRecord(GenericRecord record) { this.record = record; }

    @DynamoDbBean
    public class GenericRecord {
        private String id;
        private String createdDate;

        @DynamoDbPartitionKey
        public String getId() { return this.id; }
        public void setId(String id) { this.id = id; }

        public String getCreatedDate() { return this.createdDate; }
        public void setCreatedDate(String createdDate) { this.createdDate =
            createdDate; }
    }
}
```

Lombok

```
@Data
@DynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
```

```
    private String createdDate;  
}
```

Você pode usar a anotação de nivelamento para nivelar quantas classes elegíveis forem necessárias. As limitações a seguir aplicam-se:

- Todos os nomes de atributos devem ser exclusivos depois de serem nivelados.
- Nunca deve haver mais de uma chave de partição, chave de classificação ou nome de tabela.

Use esquemas estáticos

Ao criar um esquema de tabela estático, use o `flatten()` método do construtor. Você também fornece os métodos getter e setter que identificam a classe contida.

```
StaticTableSchema<GenericRecord> GENERIC_RECORD_SCHEMA =  
    StaticTableSchema.builder(GenericRecord.class)  
        .newItemSupplier(GenericRecord::new)  
        .addAttribute(String.class, a -> a.name("id")  
            .getter(GenericRecord::getId)  
            .setter(GenericRecord::setId)  
            .tags(primaryPartitionKey()))  
        .addAttribute(String.class, a -> a.name("created_date")  
            .getter(GenericRecord::getCreatedDate)  
            .setter(GenericRecord::setCreatedDate))  
        .build();  
  
StaticTableSchema<Customer> CUSTOMER_SCHEMA =  
    StaticTableSchema.builder(Customer.class)  
        .newItemSupplier(Customer::new)  
        .addAttribute(String.class, a -> a.name("name")  
            .getter(Customer::getName)  
            .setter(Customer::setName))  
        // Because we are flattening a component object, we supply a  
        // getter and setter so the  
        // mapper knows how to access it.  
        .flatten(GENERIC_RECORD_SCHEMA, Customer::getRecord,  
Customer::setRecord)  
        .build();
```

O exemplo anterior do esquema estático usa as seguintes classes de dados.

Classes de dados

Standard data class

```
public class Customer {  
    private String name;  
    private GenericRecord record;  
  
    public String getName() { return this.name; }  
    public void setName(String name) { this.name = name; }  
  
    public GenericRecord getRecord() { return this.record; }  
    public void setRecord(GenericRecord record) { this.record = record; }  
  
}  
  
public class GenericRecord {  
    private String id;  
    private String createdDate;  
  
    public String getId() { return this.id; }  
    public void setId(String id) { this.id = id; }  
  
    public String getCreatedDate() { return this.createdDate; }  
    public void setCreatedDate(String createdDate) { this.createdDate =  
        createdDate; }  
}
```

Lombok

```
@Data  
public class Customer {  
    private String name;  
    private GenericRecord record;  
}  
  
@Data  
public class GenericRecord {  
    private String id;  
    private String createdDate;  
}
```

Você pode usar o padrão builder para nivelar quantas classes elegíveis forem necessárias.

Implicações para outro código

Quando você usa o `@DynamoDbFlatten` atributo (ou método `flatten() builder`), o item no DynamoDB contém um atributo para cada atributo do objeto composto. Também inclui os atributos do objeto de composição.

Por outro lado, se você anotar uma classe de dados com uma classe composta e não usar `@DynamoDbFlatten`, o item será salvo com o objeto composto como um único atributo.

Por exemplo, compare a `Customer` classe mostrada no [exemplo de nivelamento com composição com](#) e sem nivelamento do atributo. `record` Você pode visualizar a diferença com o JSON, conforme mostrado na tabela a seguir.

Com achatamento	Sem achatamento
3 atributos	2 atributos
<pre>{ "id": "1", "createdDate": "today", "name": "my name" }</pre>	<pre>{ "id": "1", "record": { "createdDate": "today", "name": "my name" } }</pre>

A diferença se torna importante se você tiver outro código acessando a tabela do DynamoDB que espera encontrar determinados atributos.

Trabalhe com atributos aninhados

Um atributo aninhado no DynamoDB está incorporado em outro atributo. Os exemplos são elementos da lista e entradas do mapa.

Em Java, um atributo aninhado do DynamoDB corresponde a um membro de uma classe que é ou `List` `Map`. Também corresponde a uma instância de um tipo complexo, como `Address` ou `PhoneNumber`, conforme usado na `Person` classe a seguir.

Classe Person

```
@DynamoDbBean
```

```
public class Person {  
    Integer id;  
    String firstName;  
    String lastName;  
    Integer age;  
    Map<String, Address> addresses;  
    List<PhoneNumber> phoneNumbers;  
  
    List<String> hobbies;  
  
    @DynamoDbPartitionKey  
    public Integer getId() {  
        return id;  
    }  
  
    public void setId(Integer id) {  
        this.id = id;  
    }  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
  
    public String getLastname() {  
        return lastName;  
    }  
  
    public void setLastName(String lastName) {  
        this.lastName = lastName;  
    }  
  
    public Integer getAge() {  
        return age;  
    }  
  
    public void setAge(Integer age) {  
        this.age = age;  
    }  
  
    public Map<String, Address> getAddresses() {
```

```
        return addresses;
    }

    public void setAddresses(Map<String, Address> addresses) {
        this.addresses = addresses;
    }

    public List<PhoneNumber> getPhoneNumbers() {
        return phoneNumbers;
    }

    public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
        this.phoneNumbers = phoneNumbers;
    }

    public List<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(List<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
            "id=" + id +
            ", firstName='" + firstName + '\'' +
            ", lastName='" + lastName + '\'' +
            ", age=" + age +
            ", addresses=" + addresses +
            ", phoneNumbers=" + phoneNumbers +
            ", hobbies=" + hobbies +
            '}';
    }
}
```

Classe Address

```
@DynamoDbBean
public class Address {
    private String street;
    private String city;
```

```
private String state;
private String zipCode;

public Address() {
}

public String getStreet() {
    return this.street;
}

public String getCity() {
    return this.city;
}

public String getState() {
    return this.state;
}

public String getZipCode() {
    return this.zipCode;
}

public void setStreet(String street) {
    this.street = street;
}

public void setCity(String city) {
    this.city = city;
}

public void setState(String state) {
    this.state = state;
}

public void setZipCode(String zipCode) {
    this.zipCode = zipCode;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Address address = (Address) o;
```

```
        return Objects.equals(street, address.street) && Objects.equals(city,
address.city) && Objects.equals(state, address.state) && Objects.equals(zipCode,
address.zipCode);
    }

@Override
public int hashCode() {
    return Objects.hash(street, city, state, zipCode);
}

@Override
public String toString() {
    return "Address{" +
        "street='" + street + '\'' +
        ", city='" + city + '\'' +
        ", state='" + state + '\'' +
        ", zipCode='" + zipCode + '\'' +
        '}';
}
}
```

Classe PhoneNumber

```
@DynamoDbBean
public class PhoneNumber {
    String type;
    String number;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }
}
```

```
@Override  
public String toString() {  
    return "PhoneNumber{" +  
        "type='" + type + '\'' +  
        ", number='" + number + '\'' +  
        '}';  
}  
}
```

Mapear atributos aninhados

Use classes anotadas

Você pode salvar atributos aninhados para classes personalizadas fazendo anotações neles. A `Address` classe e a `PhoneNumber` classe mostradas anteriormente são anotadas somente com a `@DynamoDbBean` anotação. Quando a DynamoDB Enhanced Client API cria o esquema de tabela para a classe com `Person` o seguinte trecho, a API descobre o uso das classes `PhoneNumber` e `Address` e cria os mapeamentos correspondentes para funcionar com `Address` o DynamoDB.

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

Use esquemas aninhados

A abordagem alternativa é usar construtores de esquemas de tabelas estáticas para cada uma das classes, conforme mostrado no código a seguir.

Os esquemas de tabela das `PhoneNumber` classes `Address` e são abstratos no sentido de que não podem ser usados com uma tabela do DynamoDB. Isso ocorre porque eles não têm definições para a chave primária. No entanto, eles são usados como esquemas aninhados no esquema de tabela da classe. `Person`

Depois de comentar as linhas 1 e 2 na definição de `PERSON_TABLE_SCHEMA`, você vê o código que usa os esquemas de tabela abstrata. O uso de `documentOf` in `EnhanceType.documentOf(...)` método não indica que o método retorne um `EnhancedDocument` tipo da API de documentos aprimorados. O `documentOf(...)` método nesse contexto retorna um objeto que sabe como mapear seu argumento de classe de e para os atributos da tabela do DynamoDB usando o argumento do esquema da tabela.

Código de esquema estático

```
// Abstract table schema that cannot be used to work with a DynamoDB table,  
// but can be used as a nested schema.  
public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =  
TableSchema.builder(Address.class)  
    .newItemSupplier(Address::new)  
    .addAttribute(String.class, a -> a.name("street")  
        .getter(Address::getStreet)  
        .setter(Address::setStreet))  
    .addAttribute(String.class, a -> a.name("city")  
        .getter(Address::getCity)  
        .setter(Address::setCity))  
    .addAttribute(String.class, a -> a.name("zipcode")  
        .getter(Address::getZipCode)  
        .setter(Address::setZipCode))  
    .addAttribute(String.class, a -> a.name("state")  
        .getter(Address::getState)  
        .setter(Address::setState))  
    .build();  
  
// Abstract table schema that cannot be used to work with a DynamoDB table,  
// but can be used as a nested schema.  
public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =  
TableSchema.builder(PhoneNumber.class)  
    .newItemSupplier(PhoneNumber::new)  
    .addAttribute(String.class, a -> a.name("type")  
        .getter(PhoneNumber::getType)  
        .setter(PhoneNumber::setType))  
    .addAttribute(String.class, a -> a.name("number")  
        .getter(PhoneNumber::getNumber)  
        .setter(PhoneNumber::setNumber))  
    .build();  
  
// A static table schema that can be used with a DynamoDB table.  
// The table schema contains two nested schemas that are used to perform mapping  
to/from DynamoDB.  
public static final TableSchema<Person> PERSON_TABLE_SCHEMA =  
TableSchema.builder(Person.class)  
    .newItemSupplier(Person::new)  
    .addAttribute(Integer.class, a -> a.name("id")  
        .getter(Person::getId)  
        .setter(Person::setId)  
        .addTag(StaticAttributeTags.primaryPartitionKey()))
```

```

        .addAttribute(String.class, a -> a.name("firstName")
                      .getter(Person::getFirstName)
                      .setter(Person::setFirstName))
        .addAttribute(String.class, a -> a.name("lastName")
                      .getter(Person::getLastName)
                      .setter(Person::setLastName))
        .addAttribute(Integer.class, a -> a.name("age")
                      .getter(Person::getAge)
                      .setter(Person::setAge))
        .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
                      .getter(Person::getHobbies)
                      .setter(Person::setHobbies))
        .addAttribute(EnhancedType.mapOf(
            EnhancedType.of(String.class),
            // 1. Use mapping functionality of the Address table
schema.
            EnhancedType.documentOf(Address.class,
TABLE_SCHEMA_ADDRESS)), a -> a.name("addresses")
                      .getter(Person::getAddresses)
                      .setter(Person::setAddresses))
        .addAttribute(EnhancedType.listOf(
            EnhancedType.of(String.class),
            // 2. Use mapping functionality of the PhoneNumber table
schema.
            EnhancedType.documentOf(PhoneNumber.class,
TABLE_SCHEMA_PHONENUMBER)), a -> a.name("phoneNumbers")
                      .getter(Person::getPhoneNumbers)
                      .setter(Person::setPhoneNumbers))
        .build();
    
```

Atributos aninhados do projeto

Para `scan()` métodos `query()` e, você pode especificar quais atributos você deseja que sejam retornados nos resultados usando chamadas de método como `addNestedAttributeToProject()` `attributesToProject()` e. A API do DynamoDB Enhanced Client converte os parâmetros de chamada do método Java [em expressões de projeção](#) antes que a solicitação seja enviada.

O exemplo a seguir preenche a Person tabela com dois itens e, em seguida, executa três operações de varredura.

A primeira varredura acessa todos os itens da tabela para comparar os resultados com as outras operações de varredura.

A segunda varredura usa o método [addNestedAttributeToProject\(\)](#)builder para retornar somente o valor do street atributo.

A terceira operação de varredura usa o método [attributesToProject\(\)](#)builder para retornar os dados do atributo de primeiro nível,hobbies. O tipo de atributo de hobbies é uma lista. Para acessar itens individuais da lista, execute uma get() operação na lista.

```
personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
// Use a utility class to add items to the Person table.
List<Person> personList = PersonUtils.getItemsForCount(2);
// This utility method performs a put against DynamoDB to save the instances in
the list argument.
PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

// The first scan logs all items in the table to compare to the results of the
subsequent scans.
final PageIterable<Person> allItems = personDynamoDbTable.scan();
allItems.items().forEach(p ->
    // 1. Log what is in the table.
    logger.info(p.toString()));

// Scan for nested attributes.
PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
    // Use the 'addNestedAttributeToProject()' or
'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street")
    ));

streetScanResult.items().forEach(p ->
    //2. Log the results of requesting nested attributes.
    logger.info(p.toString()));

// Scan for a top-level list attribute.
PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
    // Use the 'attributesToProject()' method to access first-level
attributes.
    .attributesToProject("hobbies"));

phoneNumbersScanResult.items().forEach((p) -> {
```

```
// 3. Log the results of the request for the 'hobbies' attribute.  
logger.info(p.toString());  
// To access an item in a list, first get the parent attribute, 'hobbies',  
then access items in the list.  
String hobby = p.getHobbies().get(1);  
// 4. Log an item in the list.  
logger.info(hobby);  
});
```

```
// Logged results from comment line 1.  
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,  
addresses={work=Address{street='street 21', city='city 21', state='state 21',  
zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',  
zipCode='22222'}}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},  
PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]  
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,  
addresses={work=Address{street='street 11', city='city 11', state='state 11',  
zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',  
zipCode='11111'}}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},  
PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]  
  
// Logged results from comment line 2.  
Person{id=null, firstName='null', lastName='null', age=null,  
addresses={work=Address{street='street 21', city='null', state='null',  
zipCode='null'}}}, phoneNumbers=null, hobbies=null  
Person{id=null, firstName='null', lastName='null', age=null,  
addresses={work=Address{street='street 11', city='null', state='null',  
zipCode='null'}}}, phoneNumbers=null, hobbies=null  
  
// Logged results from comment lines 3 and 4.  
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,  
phoneNumbers=null, hobbies=[hobby 2, hobby 21]}  
hobby 21  
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,  
phoneNumbers=null, hobbies=[hobby 1, hobby 11]}  
hobby 11
```

Note

Se o `attributesToProject()` método seguir qualquer outro método do construtor que adiciona atributos que você deseja projetar, a lista de nomes de atributos fornecida ao `attributesToProject()` substitui todos os outros nomes de atributos.

Uma varredura realizada com a `ScanEnhancedRequest` instância no trecho a seguir retorna somente dados do hobby.

```
ScanEnhancedRequest lastOverwrites = ScanEnhancedRequest.builder()
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("firstName")
    // If the 'attributesToProject()' method follows other builder methods
    // that add attributes for projection,
    // its list of attributes replace all previous attributes.
    .attributesToProject("hobbies")
    .build();

PageIterable<Person> hobbiesOnlyResult =
    personDynamoDbTable.scan(lastOverwrites);
hobbiesOnlyResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
    phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
```

O trecho de código a seguir usa o `attributesToProject()` método primeiro. Essa ordenação preserva todos os outros atributos solicitados.

```
ScanEnhancedRequest attributesPreserved = ScanEnhancedRequest.builder()
    // Use 'attributesToProject()' first so that the method call does not
    // replace all other attributes
    // that you want to project.
    .attributesToProject("firstName")
    .addNestedAttributeToProject(
        NestedAttributeName.create("addresses", "work", "street"))
    .addAttributeToProject("hobbies")
    .build();

PageIterable<Person> allAttributesResult =
    personDynamoDbTable.scan(attributesPreserved);
allAttributesResult.items().forEach(p ->
    logger.info(p.toString()));

// Logged results.
```

```
Person{id=null, firstName='first name 2', lastName='null', age=null,
       addresses={work=Address{street='street 21', city='null', state='null',
                             zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
Person{id=null, firstName='first name 1', lastName='null', age=null,
       addresses={work=Address{street='street 11', city='null', state='null',
                             zipCode='null'}}, phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
```

Preserve objetos vazios com `@DynamoDbPreserveEmptyObject`

Se você salvar um bean no Amazon DynamoDB com objetos vazios e quiser que o SDK recrie os objetos vazios após a recuperação, anote o getter do bean interno com `@DynamoDbPreserveEmptyObject`.

Para ilustrar como a anotação funciona, o exemplo de código usa os dois beans a seguir.

Exemplo de feijão

A classe de dados a seguir contém dois `InnerBean` campos. O método `getter, getInnerBeanWithoutAnno()`, não é anotado com `@DynamoDbPreserveEmptyObject`. O `getInnerBeanWithAnno()` método é anotado.

```
@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
    public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
    { this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

    @DynamoDbPreserveEmptyObject
```

```
public InnerBean getInnerBeanWithAnno() { return innerBeanWithAnno; }
public void setInnerBeanWithAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithAnno = innerBeanWithAnno; }

@Override
public String toString() {
    return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
        .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
        .add("innerBeanWithAnno=" + innerBeanWithAnno)
        .add("id='" + id + "'")
        .add("name='" + name + "'")
        .toString();
}
}
```

As instâncias da InnerBean classe a seguir são campos MyBean e são inicializadas como objetos vazios no código de exemplo.

```
@DynamoDbBean
public class InnerBean {

    private String innerBeanField;

    public String getInnerBeanField() {
        return innerBeanField;
    }

    public void setInnerBeanField(String innerBeanField) {
        this.innerBeanField = innerBeanField;
    }

    @Override
    public String toString() {
        return "InnerBean{" +
            "innerBeanField='" + innerBeanField + '\'' +
            '}';
    }
}
```

O exemplo de código a seguir salva um MyBean objeto com beans internos inicializados no DynamoDB e, em seguida, recupera o item. A saída registrada mostra que o innerBeanWithoutAnno não foi inicializado, mas innerBeanWithAnno foi criado.

```
public MyBean preserveEmptyObjectUsingGetItemExample(DynamoDbTable<MyBean>
myBeanTable) {
    // Save an item to DynamoDB.
    MyBean bean = new MyBean();
    bean.setId("1");
    bean.setInnerBeanWithoutAnno(new InnerBean());      // Instantiate the inner bean.
    bean.setInnerBeanWithAnno(new InnerBean());         // Instantiate the inner bean.
    myBeanTable.putItem(bean);

    GetItemEnhancedRequest request = GetItemEnhancedRequest.builder()
        .key(Key.builder().partitionValue("1").build())
        .build();
    MyBean myBean = myBeanTable.getItem(request);

    logger.info(myBean.toString());
    // Output 'MyBean[innerBeanWithoutAnno=null,
    innerBeanWithAnno=InnerBean{innerBeanField='null'}, id='1', name='null']'.

    return myBean;
}
```

Esquema estático alternativo

Você pode usar a seguinte `StaticTableSchema` versão dos esquemas de tabela no lugar das anotações nos beans.

```
public static TableSchema<MyBean> buildStaticSchemas() {

    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =
        StaticTableSchema.builder(InnerBean.class)
            .newItemSupplier(InnerBean::new)
            .addAttribute(String.class, a -> a.name("innerBeanField")
                .getter(InnerBean::getInnerBeanField)
                .setter(InnerBean::setInnerBeanField))
            .build();

    return StaticTableSchema.builder(MyBean.class)
        .newItemSupplier(MyBean::new)
        .addAttribute(String.class, a -> a.name("id")
            .getter(MyBean::getId)
            .setter(MyBean::setId)
            .addTag(primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("name"))
```

```
        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
    a -> a.name("innerBean1")
        .getter(MyBean::getInnerBeanWithoutAnno)
        .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.preserveEmptyObject(true)),
    a -> a.name("innerBean2")
        .getter(MyBean::getInnerBeanWithAnno)
        .setter(MyBean::setInnerBeanWithAnno))
    .build());
}
```

Evite salvar atributos nulos de objetos aninhados

Você pode ignorar atributos nulos de objetos aninhados ao salvar um objeto de classe de dados no DynamoDB aplicando a anotação `@DynamoDbIgnoreNulls`. Por outro lado, atributos de nível superior com valores nulos nunca são salvos no banco de dados.

Para ilustrar como a anotação funciona, o exemplo de código usa os dois beans a seguir.

Exemplo de feijão

A classe de dados a seguir contém dois `InnerBean` campos. O método `getter, getInnerBeanWithoutAnno()`, não é anotado. O `getInnerBeanWithIgnoreNullsAnno()` método é anotado com `@DynamoDbIgnoreNulls`.

```
@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;
    private InnerBean innerBeanWithIgnoreNullsAnno;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getName() { return name; }
```

```
public void setName(String name) { this.name = name; }

public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
{ this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

@DynamoDbIgnoreNulls
public InnerBean getInnerBeanWithIgnoreNullsAnno() { return
innerBeanWithIgnoreNullsAnno; }
public void setInnerBeanWithIgnoreNullsAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithIgnoreNullsAnno = innerBeanWithAnno; }

@Override
public String toString() {
    return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
        .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
        .add("innerBeanWithIgnoreNullsAnno=" + innerBeanWithIgnoreNullsAnno)
        .add("id='" + id + "'")
        .add("name='" + name + "'")
        .toString();
}
}
```

As instâncias da InnerBean classe a seguir são campos de MyBean e são usadas no código de exemplo a seguir.

```
@DynamoDbBean
public class InnerBean {

    private String innerBeanFieldString;
    private Integer innerBeanFieldInteger;

    public String getInnerBeanFieldString() { return innerBeanFieldString; }
    public void setInnerBeanFieldString(String innerBeanFieldString)
    { this.innerBeanFieldString = innerBeanFieldString; }

    public Integer getInnerBeanFieldInteger() { return innerBeanFieldInteger; }
    public void setInnerBeanFieldInteger(Integer innerBeanFieldInteger)
    { this.innerBeanFieldInteger = innerBeanFieldInteger; }

    @Override
    public String toString() {
        return new StringJoiner(", ", InnerBean.class.getSimpleName() + "[", "]")
```

```
        .add("innerBeanFieldString=''" + innerBeanFieldString + "'")  
        .add("innerBeanFieldInteger=" + innerBeanFieldInteger)  
        .toString();  
    }  
}
```

O exemplo de código a seguir cria um InnerBean objeto e define somente um de seus dois atributos com um valor.

```
public void ignoreNullsAnnoUsingPutItemExample(DynamoDbTable<MyBean> myBeanTable) {  
    // Create an InnerBean object and give only one attribute a value.  
    InnerBean innerBeanOneAttributeSet = new InnerBean();  
    innerBeanOneAttributeSet.setInnerBeanFieldInteger(200);  
  
    // Create a MyBean instance and use the same InnerBean instance both for  
    attributes.  
    MyBean bean = new MyBean();  
    bean.setId("1");  
    bean.setInnerBeanWithoutAnno(innerBeanOneAttributeSet);  
    bean.setInnerBeanWithIgnoreNullsAnno(innerBeanOneAttributeSet);  
  
    Map<String, AttributeValue> itemMap = myBeanTable.tableSchema().itemToMap(bean,  
true);  
    logger.info(itemMap.toString());  
    // Log the map that is sent to the database.  
    //  
    {innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}),  
id=AttributeValue(S=1),  
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),  
innerBeanFieldString=AttributeValue(NUL=true)})}  
  
    // Save the MyBean object to the table.  
    myBeanTable.putItem(bean);  
}
```

Para visualizar os dados de baixo nível enviados ao DynamoDB, o código registra o mapa de atributos antes de salvar o objeto MyBean

A saída registrada mostra que innerBeanWithIgnoreNullsAnno gera um atributo,

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

A `innerBeanWithoutAnno` instância gera dois atributos. Um atributo tem um valor de 200 e o outro é um atributo de valor nulo.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),  
innerBeanFieldString=AttributeValue(NUL=true)})
```

Representação JSON do mapa de atributos

A representação JSON a seguir facilita a visualização dos dados salvos no DynamoDB.

```
{  
    "id": {  
        "S": "1"  
    },  
    "innerBeanWithIgnoreNullsAnno": {  
        "M": {  
            "innerBeanFieldInteger": {  
                "N": "200"  
            }  
        }  
    },  
    "innerBeanWithoutAnno": {  
        "M": {  
            "innerBeanFieldInteger": {  
                "N": "200"  
            },  
            "innerBeanFieldString": {  
                "NULL": true  
            }  
        }  
    }  
}
```

Esquema estático alternativo

Você pode usar a seguinte `StaticTableSchema` versão dos esquemas de tabela nas anotações da classe de dados no local.

```
public static TableSchema<MyBean> buildStaticSchemas() {  
  
    StaticTableSchema<InnerBean> innerBeanStaticTableSchema =  
        StaticTableSchema.builder(InnerBean.class)
```

```
.newItemSupplier(InnerBean::new)
.addAttribute(String.class, a -> a.name("innerBeanFieldString")
    .getter(InnerBean::getInnerBeanFieldString)
    .setter(InnerBean::setInnerBeanFieldString))
.addAttribute(Integer.class, a -> a.name("innerBeanFieldInteger")
    .getter(InnerBean::getInnerBeanFieldInteger)
    .setter(InnerBean::setInnerBeanFieldInteger))
.build();

return StaticTableSchema.builder(MyBean.class)
    .newItemSupplier(MyBean::new)
    .addAttribute(String.class, a -> a.name("id")
        .getter(MyBean::getId)
        .setter(MyBean::setId)
        .addTag(primaryPartitionKey())))
    .addAttribute(String.class, a -> a.name("name")
        .getter(MyBean::getName)
        .setter(MyBean::setName))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema),
        a -> a.name("innerBeanWithoutAnno")
            .getter(MyBean::getInnerBeanWithoutAnno)
            .setter(MyBean::setInnerBeanWithoutAnno))
    .addAttribute(EnhancedType.documentOf(InnerBean.class,
        innerBeanStaticTableSchema,
        b -> b.ignoreNulls(true)),
        a -> a.name("innerBeanWithIgnoreNullsAnno")
            .getter(MyBean::getInnerBeanWithIgnoreNullsAnno)
            .setter(MyBean::setInnerBeanWithIgnoreNullsAnno))
    .build();
}
```

API de documentos aprimorada para DynamoDB

A [API de documentos aprimorada](#) para foi AWS SDK for Java 2.x projetada para funcionar com dados orientados a documentos que não têm um esquema fixo. No entanto, ele também permite que você use classes personalizadas para mapear atributos individuais.

A API de documentos aprimorada é a sucessora da [API de documentos](#) da AWS SDK for Java v1.x.

Sumário

- [Etapas preliminares para trabalhar com a API de documentos aprimorada](#)

- [Crie um DocumentTableSchema e um DynamoDbTable](#)
- [Crie documentos aprimorados](#)
 - [Crie a partir de uma string JSON](#)
 - [Crie a partir de elementos individuais](#)
- [Execute operações CRUD](#)
- [Acesse atributos aprimorados do documento como objetos personalizados](#)
- [Use e EnhancedDocument sem o DynamoDB](#)

Etapas preliminares para trabalhar com a API de documentos aprimorada

A Enhanced Document API exige as mesmas [dependências](#) que são necessárias para a API do DynamoDB Enhanced Client. Também requer uma [DynamoDbEnhancedClientinstância](#), conforme mostrado no início deste tópico.

Como a API de documentos aprimorada foi lançada com a versão 2.20.3 do AWS SDK for Java 2.x, você precisa dessa versão ou superior.

Crie um **DocumentTableSchema** e um **DynamoDbTable**

Para invocar comandos em uma tabela do DynamoDB usando a Enhanced Document API, associe a tabela a um objeto de recurso < > do lado do [DynamoDbTablecliente EnhancedDocument](#).

O `table()` método aprimorado do cliente cria uma `DynamoDbTable<EnhancedDocument>` instância e exige parâmetros para o nome da tabela do DynamoDB e uma `DocumentTableSchema`

O construtor de um [DocumentTableSchema](#) requer uma chave de índice primária e um ou mais provedores de conversão de atributos. O `AttributeConverterProvider.defaultProvider()` método fornece conversores para [tipos padrão](#). Ele deve ser especificado mesmo se você fornecer um provedor de conversão de atributos personalizado. Você pode adicionar uma chave de índice secundária opcional ao construtor.

O trecho de código a seguir mostra o código que gera a representação do lado do cliente de uma tabela do DynamoDB que armazena objetos sem esquema. `person EnhancedDocument`

```
DynamoDbTable<EnhancedDocument> documentDynamoDbTable =  
    enhancedClient.table("person",  
        TableSchema.documentSchemaBuilder()
```

```
// Specify the primary key attributes.  
  
.addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S)  
    .addIndexSortKey(TableMetadata.primaryIndexName(),  
"lastName", AttributeValueType.S)  
        // Specify attribute converter providers. Minimally add the  
default one.  
  
.attributeConverterProviders(AttributeConverterProvider.defaultProvider())  
    .build());  
  
// Call documentTable.createTable() if "person" does not exist in DynamoDB.  
// createTable() should be called only one time.
```

A seguir, mostramos a representação JSON de um person objeto usado nesta seção.

Objeto JSON **person**

```
{  
    "id": 1,  
    "firstName": "Richard",  
    "lastName": "Roe",  
    "age": 25,  
    "addresses":  
    {  
        "home": {  
            "zipCode": "00000",  
            "city": "Any Town",  
            "state": "FL",  
            "street": "123 Any Street"  
        },  
        "work": {  
            "zipCode": "00001",  
            "city": "Anywhere",  
            "state": "FL",  
            "street": "100 Main Street"  
        }  
    },  
    "hobbies": [  
        "Hobby 1",  
        "Hobby 2"  
    ],  
    "phoneNumbers": [  
        {
```

```
        "type": "Home",
        "number": "555-0100"
    },
    {
        "type": "Work",
        "number": "555-0119"
    }
]
```

Crie documentos aprimorados

An [EnhancedDocument](#) representa um objeto do tipo documento que tem uma estrutura complexa com atributos aninhados. An EnhancedDocument requer atributos de nível superior que correspondam aos atributos da chave primária especificados para o `DocumentTableSchema`. O conteúdo restante é arbitrário e pode consistir em atributos de nível superior e também em atributos profundamente aninhados.

Você cria uma EnhancedDocument instância usando um construtor que fornece várias maneiras de adicionar elementos.

Crie a partir de uma string JSON

Com uma string JSON, você pode criar uma chamada de método EnhancedDocument em um. O trecho a seguir cria um EnhancedDocument partir de uma string JSON retornada pelo `jsonPerson()` método auxiliar. O `jsonPerson()` método retorna a versão da string JSON do [objeto pessoal](#) mostrado anteriormente.

```
EnhancedDocument document =
    EnhancedDocument.builder()
        .json( jsonPerson() )
        .build();
```

Crie a partir de elementos individuais

Como alternativa, você pode criar uma EnhancedDocument instância a partir de componentes individuais usando métodos seguros de tipo do construtor.

O exemplo a seguir cria um documento person aprimorado semelhante ao documento aprimorado criado a partir da string JSON no exemplo anterior.

```
/* Define the shape of an address map whose JSON representation looks like the
following.
    Use 'addressMapEnhancedType' in the following EnhancedDocument.builder() to
simplify the code.
    "home": {
        "zipCode": "00000",
        "city": "Any Town",
        "state": "FL",
        "street": "123 Any Street"
    } */
EnhancedType<Map<String, String>> addressMapEnhancedType =
    EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class));

// Use the builder's typesafe methods to add elements to the enhanced
document.
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
    /* Add the map of addresses whose JSON representation looks like the
following.
    {
        "home": {
            "zipCode": "00000",
            "city": "Any Town",
            "state": "FL",
            "street": "123 Any Street"
        }
    } */
    .putMap("addresses", getAddresses(), EnhancedType.of(String.class),
addressMapEnhancedType)
    .putList("hobbies", List.of("Theater", "Golf"),
EnhancedType.of(String.class))
    .build();
```

Métodos auxiliares

```
private static String phoneNumbersJSONString() {
```

```
        return "[" +
            "{" +
                "\"type\": \"Home\", " +
                "\"number\": \"555-0140\" " +
            }, " +
            "{" +
                "\"type\": \"Work\", " +
                "\"number\": \"555-0155\" " +
            }" +
        "]";
    }

private static Map<String, Map<String, String>> getAddresses() {
    return Map.of(
        "home", Map.of(
            "zipCode", "00002",
            "city", "Any Town",
            "state", "ME",
            "street", "123 Any Street"));
}
```

Execute operações CRUD

Depois de definir uma EnhancedDocument instância, você pode salvá-la em uma tabela do DynamoDB. O trecho de código a seguir usa o [PersonDocument](#) que foi criado a partir de elementos individuais.

```
documentDynamoDbTable.putItem(personDocument);
```

Depois de ler uma instância de documento aprimorada do DynamoDB, você pode extrair os valores dos atributos individuais usando getters, conforme mostrado no trecho de código a seguir, que acessam os dados salvos do `personDocument`. Como alternativa, você pode extrair o conteúdo completo em uma string JSON, conforme mostrado na última parte do código de exemplo.

```
// Read the item.
EnhancedDocument personDocFromDb =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).build());

// Access top-level attributes.
logger.info("Name: {} {}", personDocFromDb.getString("firstName"),
personDocFromDb.getString("lastName"));
```

```
// Name: Shirley Rodriguez

    // Typesafe access of a deeply nested attribute. The addressMapEnhancedType
    shown previously defines the shape of an addresses map.
    Map<String, Map<String, String>> addresses =
personDocFromDb.getMap("addresses", EnhancedType.of(String.class),
addressMapEnhancedType);
    addresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));
    // {zipCode=00002, city=Any Town, street=123 Any Street, state=ME}

    // Alternatively, work with AttributeValue types checking along the way for
deeply nested attributes.
    Map<String, AttributeValue> addressesMap =
personDocFromDb.getMapOfUnknownType("addresses");
    addressesMap.keySet().forEach((String k) -> {
        logger.info("Looking at data for [{}] address", k);
        // Looking at data for [home] address
        AttributeValue value = addressesMap.get(k);
        AttributeValue cityValue = value.m().get("city");
        if (cityValue != null) {
            logger.info(cityValue.s());
            // Any Town
        }
    });
}

List<AttributeValue> phoneNumbers =
personDocFromDb.getListOfUnknownType("phoneNumbers");
    phoneNumbers.forEach((AttributeValue av) -> {
        if (av.hasM()) {
            AttributeValue type = av.m().get("type");
            if (type.s() != null) {
                logger.info("Type of phone: {}", type.s());
                // Type of phone: Home
                // Type of phone: Work
            }
        }
    });
}

String jsonPerson = personDocFromDb.toJson();
logger.info(jsonPerson);
// {"firstName":"Shirley","lastName":"Rodriguez","addresses": {"home": {"zipCode": "00002", "city": "Any Town", "street": "123 Any Street", "state": "ME"}}, "hobbies": ["Theater", "Golf"],
```

```
//      "id":50,"nullAttribute":null,"age":53,"phoneNumbers":  
[{"number":"555-0140","type":"Home"}, {"number":"555-0155","type":"Work"}]]
```

EnhancedDocumentas instâncias podem ser usadas com qualquer método de [DynamoDbTable](#) ou [DynamoDbEnhancedClient](#) no lugar de classes de dados mapeadas.

Acesse atributos aprimorados do documento como objetos personalizados

Além de fornecer uma API para ler e gravar atributos com estruturas sem esquemas, a API de documentos aprimorados permite converter atributos de e para instâncias de classes personalizadas.

A Enhanced Document API usa AttributeConverterProviders e AttributeConverters que foram mostrados na seção de [conversão de atributos de controle](#) como parte da API DynamoDB Enhanced Client.

No exemplo a seguir, usamos a CustomAttributeConverterProvider com sua AddressConverter classe aninhada para converter Address objetos.

Este exemplo mostra que você pode misturar dados de classes e também dados de estruturas que são criadas conforme necessário. Esse exemplo também mostra que classes personalizadas podem ser usadas em qualquer nível de uma estrutura aninhada. Os Address objetos neste exemplo são valores usados em um mapa.

```
public static void attributeToAddressClassMappingExample(DynamoDbEnhancedClient  
enhancedClient, DynamoDbClient standardClient) {  
    String tableName = "customer";  
  
    // Define the DynamoDbTable for an enhanced document.  
    // The schema builder provides methods for attribute converter providers and  
    keys.  
    DynamoDbTable<EnhancedDocument> documentDynamoDbTable =  
enhancedClient.table(tableName,  
        DocumentTableSchema.builder()  
            // Add the CustomAttributeConverterProvider along with the  
            default when you build the table schema.  
            .attributeConverterProviders(  
                List.of(  
                    new CustomAttributeConverterProvider(),  
                    AttributeConverterProvider.defaultProvider()))  
            .addIndexPartitionKey(TableMetadata.primaryIndexName(), "id",  
AttributeValueType.N)
```

```
        .addIndexSortKey(TableMetadata.primaryIndexName(), "lastName",
AttributeValueType.S)
        .build());
// Create the DynamoDB table if needed.
documentDynamoDbTable.createTable();
waitForTableCreation(tableName, standardClient);

// The getAddressesForCustomMappingExample() helper method that provides
'addresses' shows the use of a custom Address class
// rather than using a Map<String, Map<String, String> to hold the address
data.
Map<String, Address> addresses = getAddressesForCustomMappingExample();

// Build an EnhancedDocument instance to save an item with a mix of structures
defined as needed and static classes.
EnhancedDocument personDocument = EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .putString("lastName", "Rodriguez")
    .putNumber("age", 53)
    .putNull("nullAttribute")
    .putJson("phoneNumbers", phoneNumbersJSONString())
    // Note the use of 'EnhancedType.of(Address.class)' instead of the more
generic
    // 'EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class))' that was used in a previous example.
    .putMap("addresses", addresses, EnhancedType.of(String.class),
EnhancedType.of(Address.class))
    .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
EnhancedType.of(String.class))
    .build();
// Save the item to DynamoDB.
documentDynamoDbTable.putItem(personDocument);

// Retrieve the item just saved.
EnhancedDocument srPerson =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build())

// Access the addresses attribute.
Map<String, Address> srAddresses = srPerson.get("addresses",
    EnhancedType.mapOf(EnhancedType.of(String.class),
    EnhancedType.of(Address.class)));
```

```
srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

documentDynamoDbTable.deleteTable();

// The content logged to the console shows that the saved maps were converted to
// Address instances.
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}
```

CustomAttributeConverterProvider código

```
public class CustomAttributeConverterProvider implements AttributeConverterProvider {

    private final Map<EnhancedType<?>, AttributeConverter<?>> converterCache =
        ImmutableMap.of(
            // 1. Add AddressConverter to the internal cache.
            EnhancedType.of(Address.class), new AddressConverter());

    public static CustomAttributeConverterProvider create() {
        return new CustomAttributeConverterProvider();
    }

    // 2. The enhanced client queries the provider for attribute converters if it
    //     encounters a type that it does not know how to convert.
    @SuppressWarnings("unchecked")
    @Override
    public <T> AttributeConverter<T> converterFor(EnhancedType<T> enhancedType) {
        return (AttributeConverter<T>) converterCache.get(enhancedType);
    }

    // 3. Custom attribute converter
    private class AddressConverter implements AttributeConverter<Address> {
        // 4. Transform an Address object into a DynamoDB map.
        @Override
        public AttributeValue transformFrom(Address address) {

            Map<String, AttributeValue> attributeValueMap = Map.of(
                "street", AttributeValue.fromS(address.getStreet()),
                "city", AttributeValue.fromS(address.getCity()),
                "state", AttributeValue.fromS(address.getState()),
                "zipCode", AttributeValue.fromS(address.getZipCode()));

            return AttributeValue.fromM(attributeValueMap);
        }
    }
}
```

```
}

// 5. Transform the DynamoDB map attribute to an Address object.
@Override
public Address transformTo(AttributeValue attributeValue) {
    Map<String, AttributeValue> m = attributeValue.m();
    Address address = new Address();
    address.setStreet(m.get("street").s());
    address.setCity(m.get("city").s());
    address.setState(m.get("state").s());
    address.setZipCode(m.get("zipCode").s());

    return address;
}

@Override
public EnhancedType<Address> type() {
    return EnhancedType.of(Address.class);
}

@Override
public AttributeValueType attributeValueType() {
    return AttributeValueType.M;
}
}

}
```

Classe Address

```
public class Address {
    private String street;
    private String city;
    private String state;
    private String zipCode;

    public Address() {
    }

    public String getStreet() {
        return this.street;
    }

    public String getCity() {
```

```
        return this.city;
    }

    public String getState() {
        return this.state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }
}
```

Método auxiliar que fornece endereços

O método auxiliar a seguir fornece o mapa que usa Address instâncias personalizadas para valores em vez de Map<String, String> instâncias genéricas para valores.

```
private static Map<String, Address> getAddressesForCustomMappingExample() {
    Address homeAddress = new Address();
    homeAddress.setStreet("100 Any Street");
    homeAddress.setCity("Any Town");
    homeAddress.setState("NC");
    homeAddress.setZipCode("00000");

    Address workAddress = new Address();
    workAddress.setStreet("123 Main Street");
    workAddress.setCity("Any Town");
```

```
    workAddress.setState("NC");
    workAddress.setZipCode("00000");

    return Map.of("home", homeAddress,
                  "work", workAddress);
}
```

Use e EnhancedDocument sem o DynamoDB

Embora você geralmente use uma instância de an EnhancedDocument para ler e gravar itens do tipo documento do DynamoDB, ela também pode ser usada independentemente do DynamoDB.

Você pode usar a capacidade deles EnhancedDocuments de converter entre cadeias de caracteres JSON ou objetos personalizados em mapas de baixo nível, AttributeValues conforme mostrado no exemplo a seguir.

```
public static void conversionWithoutDynamoDbExample() {
    Address address = new Address();
    address.setCity("my city");
    address.setState("my state");
    address.setStreet("my street");
    address.setZipCode("00000");

    // Build an EnhancedDocument instance for its conversion functionality alone.
    EnhancedDocument addressEnhancedDoc = EnhancedDocument.builder()
        // Important: You must specify attribute converter providers when you
        // build an EnhancedDocument instance not used with a DynamoDB table.
        .attributeConverterProviders(new CustomAttributeConverterProvider(),
DefaultAttributeConverterProvider.create())
        .put("addressDoc", address, Address.class)
        .build();

    // Convert address to a low-level item representation.
    final Map<String, AttributeValue> addressAsAttributeMap =
addressEnhancedDoc.getMapOfUnknownType("addressDoc");
    logger.info("addressAsAttributeMap: {}", addressAsAttributeMap.toString());

    // Convert address to a JSON string.
    String addressAsJsonString = addressEnhancedDoc.getJson("addressDoc");
    logger.info("addressAsJsonString: {}", addressAsJsonString);
    // Convert addressEnhancedDoc back to an Address instance.
    Address addressConverted = addressEnhancedDoc.get("addressDoc",
Address.class);
```

```
        logger.info("addressConverted: {}", addressConverted.toString());
    }

    /* Console output:
       addressAsAttributeMap: {zipCode=AttributeValue(S=00000),
       state=AttributeValue(S=my state), street=AttributeValue(S=my street),
       city=AttributeValue(S=my city)}
       addressAsString: {"zipCode":"00000","state":"my state","street":"my
       street","city":"my city"}
       addressConverted: Address{street='my street', city='my city', state='my
       state', zipCode='00000'}
    */
}
```

Note

Ao usar um documento aprimorado independente de uma tabela do DynamoDB, certifique-se de definir explicitamente os provedores de conversão de atributos no construtor. Por outro lado, o esquema da tabela de documentos fornece os provedores de conversão quando um documento aprimorado é usado com uma tabela do DynamoDB.

Operações assíncronas sem bloqueio

Se seu aplicativo exigir chamadas assíncronas e sem bloqueio para o DynamoDB, você poderá usar o [DynamoDbEnhancedAsyncClient](#). É semelhante à implementação síncrona, mas com as seguintes diferenças principais:

1. Ao criar o `DynamoDbEnhancedAsyncClient`, você deve usar a versão assíncrona do cliente padrão, `DynamoDbAsyncClient`, conforme mostrado no trecho a seguir.

```
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dynamoDbAsyncClient)
        .build();
```

2. Os métodos que retornam um único objeto de dados retornam um `CompletableFuture` do resultado em vez de somente o resultado. Seu aplicativo pode então fazer outro trabalho sem precisar bloquear o resultado. O trecho a seguir mostra o método `getItem()` assíncrono.

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
```

```
// Perform other work here.
return result.join(); // Now block and wait for the result.
```

3. Métodos que retornam listas paginadas de resultados retornam um [SdkPublisher](#) em vez de um [SdkIterable](#) que o síncrono DynamoDbEnhanceClient retorna para os mesmos métodos. Seu aplicativo pode então inscrever um manipulador nesse editor para lidar com os resultados de forma assíncrona, sem precisar bloquear.

```
PagePublisher<Customer> results = customerDynamoDbTable.query(r ->
    r.queryConditional(keyEqualTo(k -> k.partitionValue("Smith"))));
results.subscribe(myCustomerResultsProcessor);
// Perform other work and let the processor handle the results asynchronously.
```

Para obter um exemplo mais completo de como trabalhar com o `SdkPublisher` API, consulte [o exemplo](#) na seção que discute o `scan()` método assíncrono deste guia.

Anotações de classes de dados

A tabela a seguir lista as anotações que podem ser usadas em classes de dados e fornece links para informações e exemplos neste guia.

Anotações de classe de dados usadas neste guia

Anotação	O que ele faz	Onde é mostrado neste guia
DynamoDbAttribute	Define ou renomeia uma propriedade de bean que é mapeada para um atributo de tabela do DynamoDB.	<ul style="list-style-type: none"> Discussão inicial. Seção de introdução — consulte Nota. Exemplos do método InQuery em MovieActor classe.
DynamoDbBean	Marca uma classe de dados como mapeável para um esquema de tabela.	Primeiro uso na classe Customer na seção Primeiros passos. Vários usos aparecem em todo o guia.

Anotação	O que ele faz	Onde é mostrado neste guia
DynamoDbConvertedBy	Associa um personalizado ao <code>AttributeConverter</code> atributo anotado.	Discussão inicial e exemplo.
DynamoDbFlatten	Nivela todos os atributos de uma classe de dados separada do DynamoDB e os adiciona como atributos de nível superior ao registro que é lido e gravado no banco de dados.	<ul style="list-style-type: none"> Discussão inicial. Implicações para outro código.
DynamoDbIgnore	Faz com que o atributo permaneça não mapeado.	<ul style="list-style-type: none"> Discussão inicial. Use na ProductCatalog aula.
DynamoDbIgnoreNulls	Impede salvar atributos nulos de objetos aninhados DynamoDb .	Discussão e exemplos.
DynamoDbImmutable	Marca uma classe de dados imutável como mapeável para um esquema de tabela.	<ul style="list-style-type: none"> Introdução à anotação. Use na ProductCatalog aula. Use com Lombok.
DynamoDbPartitionKey	Marca um atributo como a chave de partição primária (chave de hash) da DynamoDb tabela.	<ul style="list-style-type: none"> Uso inicial na classe Customer na seção Primeiros passos. Com Lombok.

Anotação	O que ele faz	Onde é mostrado neste guia
DynamoDbPreserveEmptyObject	Especifica que, se nenhum dado estiver presente para o objeto mapeado para o atributo anotado, o objeto deverá ser inicializado com todos os campos nulos.	Discussão e exemplos.
DynamoDbSecondaryPartitionKey	Marca um atributo como uma chave de partição para um índice secundário global.	<ul style="list-style-type: none"> • Use em índices secundários e exemplos. • Em exemplos do método Query. • No exemplo de Lombok • Com classes imutáveis.
DynamoDbSecondarySortKey	Marca um atributo como uma chave de classificação opcional para um índice secundário global ou local.	<ul style="list-style-type: none"> • Use em índices secundários e exemplos. • Em exemplos do método Query. • No exemplo de Lombok. • Com classes imutáveis.
DynamoDbSortKey	Marca um atributo como a chave de classificação primária opcional (chave de intervalo).	<ul style="list-style-type: none"> • Seção de introdução na classe Customer. • Com classes imutáveis. • No exemplo de Lombok. • Em exemplos do método Query.
DynamoDbUpdateBehavior	Especifica o comportamento quando esse atributo é atualizado como parte de uma operação de 'atualização', como UpdateItem.	Introdução e exemplo.

Trabalhando com HTTP/2 no AWS SDK for Java

HTTP/2 é uma grande revisão do protocolo HTTP. Esta nova versão tem vários aprimoramentos para melhorar o desempenho:

- A codificação de dados binários proporciona uma transferência de dados mais eficiente.
- A compactação de cabeçalho reduz a sobrecarga de bytes baixados pelo cliente, ajudando a obter o conteúdo para o cliente mais cedo. Isso é especialmente útil para clientes móveis que já tenham restrição na largura de banda.
- A comunicação assíncrona bidirecional (multiplexação) permite que várias solicitações e mensagens de resposta entre o cliente estejam em andamento AWS ao mesmo tempo em uma única conexão, em vez de em várias conexões, o que melhora o desempenho.

Os desenvolvedores atualizando para os SDKs mais recentes usarão automaticamente HTTP/2 quando ele for compatível com o serviço com que estiverem trabalhando. Novas interfaces de programação aproveitam perfeitamente os recursos do HTTP/2 e fornecem novas maneiras de criar aplicativos.

O AWS SDK for Java 2.x apresenta novas APIs para streaming de eventos que implementam o protocolo HTTP/2. Para ver exemplos de como usar essas novas APIs, consulte [Trabalhando com o Kinesis](#).

Aactive as métricas do SDK para o AWS SDK for Java

Com o AWS SDK for Java 2.x, você pode coletar métricas sobre os clientes de serviço em seu aplicativo, analisar a saída e, em seguida, Amazon CloudWatch, agir de acordo com ela.

Por padrão, a coleta de métricas está desativada no SDK. Este tópico ajuda você a habilitá-lo e configurá-lo.

Pré-requisitos

Antes de ativar e usar métricas, você deve concluir as seguintes etapas:

- Siga as etapas em [Configuração](#).
- Configure as dependências do seu projeto (por exemplo, no seu `build.gradle` arquivo `pom.xml` ou) para usar a versão 2.14.0 ou posterior do AWS SDK for Java.

Para permitir a publicação de métricas em CloudWatch, inclua também o `cloudwatch-metric-publisher` ArtifactID com o 2.14.0 número da versão ou posterior nas dependências do seu projeto.

Por exemplo:

```
<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>
```

Como ativar a coleta de métricas

Você pode habilitar métricas em seu aplicativo para um cliente de serviço ou em solicitações individuais.

Ativar métricas para uma solicitação específica

O trecho de código a seguir mostra como habilitar o editor de CloudWatch métricas para uma solicitação para Amazon DynamoDB Ele usa a configuração padrão do editor de métricas.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
DynamoDbClient ddb = DynamoDbClient.create();
ddb.listTables(ListTablesRequest.builder()
```

```
.overrideConfiguration(c -> c.addMetricPublisher(metricsPub))  
.build());
```

Ativar métricas para um cliente de serviço específico

O trecho de código a seguir mostra como habilitar um editor de CloudWatch métricas com configurações padrão para um cliente de serviço.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();  
  
DynamoDbClient ddb = DynamoDbClient.builder()  
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))  
    .build();
```

O trecho a seguir demonstra como usar uma configuração personalizada para o editor de métricas de um cliente de serviço específico. As personalizações incluem carregar um perfil de credenciais específico, especificar uma região diferente da do cliente de serviço e personalizar a frequência com que o editor envia métricas. 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();
```

Quais informações são coletadas?

A coleção de métricas inclui o seguinte:

- Número de solicitações de API, incluindo se elas são bem-sucedidas ou falhadas
- Informações sobre os AWS serviços que você chama em suas solicitações de API, incluindo exceções retornadas
- A duração de várias operações, como empacotamento, assinatura e solicitações HTTP

- Métricas do cliente HTTP, como o número de conexões abertas, o número de solicitações pendentes e o nome do cliente HTTP usado

 Note

As métricas disponíveis variam de acordo com o cliente HTTP.

Para obter uma lista completa, consulte [Métricas do cliente de serviço](#).

Como posso usar essas informações?

Você pode usar as métricas que o SDK coleta para monitorar os clientes de serviço em seu aplicativo. Você pode analisar as tendências gerais de uso, identificar anomalias, analisar as exceções retornadas pelos clientes do serviço ou se aprofundar para entender um problema específico. Usando Amazon CloudWatch, você também pode criar alarmes para notificá-lo assim que seu aplicativo atingir uma condição definida por você.

Para obter mais informações, consulte [Usando Amazon CloudWatch métricas](#) e [usando Amazon CloudWatch alarmes](#) no [Guia do Amazon CloudWatch usuário](#).

Métricas do cliente de serviço

Com o AWS SDK for Java 2.x, você pode coletar métricas dos clientes de serviço em seu aplicativo e depois publicar (gerar) essas métricas [na Amazon CloudWatch](#).

Essas tabelas listam as métricas que você pode coletar e qualquer requisito de uso do cliente HTTP.

Para obter mais informações sobre como habilitar e configurar métricas para o SDK, consulte [Habilitando métricas do SDK](#).

Os termos usados nas tabelas significam:

- Apache: o cliente HTTP baseado em Apache () [ApacheHttpClient](#)
- Netty: o cliente HTTP baseado em Netty () [NettyNioAsyncHttpClient](#)
- CRT: o cliente HTTP AWS baseado em CRT () [AwsCrtAsyncHttpClient](#)
- Qualquer: a coleta de dados métricos não depende do cliente HTTP; isso inclui o uso do cliente HTTP baseado em URLConnection () [UrlConnectionHttpClient](#)

Métricas coletadas com cada solicitação

Nome da métrica	Descrição	Tipo	É necessário um cliente HTTP
ApiCallDuration	O tempo total necessário para concluir uma solicitação (incluindo todas as novas tentativas)	Duração	Quaisquer
ApiCallSuccessful	Verdadeiro se a chamada da API for bem-sucedida; falso se não for	Booliano	Quaisquer
CredentialsFetchDuration	O tempo necessário para obter as credenciais de AWS assinatura da solicitação	Duração	Quaisquer
MarshallingDuration	O tempo necessário para organizar a solicitação	Duração	Quaisquer
OperationName	O nome da AWS API para a qual a solicitação é feita	Segmento	Quaisquer
RetryCount	Número de vezes que o SDK tentou novamente a chamada de API	Inteiro	Quaisquer
ServiceId	ID de serviço do AWS service (Serviço da	Segmento	Quaisquer

Nome da métrica	Descrição	Tipo	É necessário um cliente HTTP
	AWS) qual a solicitação de API é feita		
TokenFetchDuration	O tempo necessário para buscar as credenciais de assinatura do token para a solicitação	Duração	Quaisquer

Métricas coletadas para cada tentativa de solicitação

Cada chamada de API pode exigir várias tentativas antes que uma resposta seja recebida. Essas métricas são coletadas para cada tentativa.

Nome da métrica	Descrição	Tipo	É necessário um cliente HTTP
AvailableConcurrency	O número de solicitações simultâneas restantes que podem ser suportadas pelo cliente HTTP sem a necessidade de estabelecer outra conexão	Inteiro	Apache, Netty, CRT
AwsExtendedRequestId	O ID da solicitação estendida da solicitação de serviço	Segmento	Quaisquer
AwsRequestId	O ID da solicitação de serviço	Segmento	Quaisquer
BackoffDelayDuration	O tempo que o SDK esperou antes dessa	Duração	Quaisquer

Nome da métrica	Descrição	Tipo	É necessário um cliente HTTP
	tentativa de chamada de API		
ConcurrencyAcquire Duration	O tempo necessário para adquirir um canal do pool de conexões	Duração	Apache, Netty, CRT
HttpClientName	O nome do HTTP que está sendo usado para a solicitação	Segmento	Apache, Netty, CRT
HttpStatusCode	O código de status retornado com a resposta HTTP	Inteiro	Quaisquer
LeasedConcurrency	O número de solicitações que estão sendo executadas atualmente pelo cliente HTTP	Inteiro	Apache, Netty, CRT
LocalStreamWindowSize	O tamanho da janela HTTP/2 local em bytes para o fluxo em que essa solicitação foi executada	Inteiro	Netty
MarshallingDuration	O tempo necessário para agrupar uma solicitação de SDK em uma solicitação HTTP	Duração	Quaisquer
MaxConcurrency	O número máximo de solicitações simultâneas suportadas pelo cliente HTTP	Inteiro	Apache, Netty, CRT

Nome da métrica	Descrição	Tipo	É necessário um cliente HTTP
PendingConcurrencyAcquires	O número de solicitações que estão bloqueadas, aguardando a disponibilidade de outra conexão TCP ou de um novo fluxo no pool de conexões	Inteiro	Apache, Netty, CRT
RemoteStreamWindowSize	O tamanho da janela HTTP/2 remota em bytes para o fluxo em que essa solicitação foi executada	Inteiro	Netty
ServiceCallDuration	O tempo necessário para se conectar ao serviço, enviar a solicitação e receber o código de status HTTP e o cabeçalho da resposta	Duração	Quaisquer
SigningDuration	O tempo necessário para assinar a solicitação HTTP	Duração	Quaisquer
UnmarshallingDuration	O tempo necessário para desorganizar uma resposta HTTP a uma resposta do SDK	Duração	Quaisquer

Recuperando resultados paginados usando o 2.x AWS SDK for Java

Muitas AWS operações retornam resultados paginados quando o objeto de resposta é muito grande para ser retornado em uma única resposta. Na AWS SDK for Java versão 1.0, a resposta contém um token que você usa para recuperar a próxima página de resultados. Por outro lado, o AWS SDK for Java 2.x tem métodos de autopaginação que fazem várias chamadas de serviço para obter automaticamente a próxima página de resultados para você. Você precisa somente escrever um código que processa os resultados. A autopaginação está disponível para clientes síncronos e assíncronos.

 Note

Esses trechos de código pressupõem que você entenda os conceitos básicos do uso do SDK e tenha configurado seu ambiente com acesso de login único.

Paginação síncrona

Os exemplos a seguir demonstram métodos de paginação síncrona para listar objetos em um bucket. Amazon S3

Itere sobre as páginas

O primeiro exemplo demonstra o uso de um objeto `listRes` paginador, uma instância [ListObjectsV2Iterable](#), para percorrer todas as páginas de resposta com o método `stream`. O código é transmitido pelas páginas de resposta, converte o fluxo de resposta em um fluxo de conteúdo do [S3Object](#) e, em seguida, processa o conteúdo do objeto. Amazon S3

As importações a seguir se aplicam a todos os exemplos nesta seção de paginação síncrona.

Importações

```
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;
```

```
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()));
```

Veja o [exemplo completo](#) em GitHub.

Itere sobre objetos

Os exemplos a seguir mostram maneiras de iterar sobre os objetos retornados na resposta e não nas páginas de resposta. O `contents` método da `ListObjectsV2Iterable` classe retorna um [SdkIterable](#) que fornece vários métodos para processar os elementos do conteúdo subjacente.

Use um stream

O trecho a seguir usa o `stream` método no conteúdo da resposta para iterar sobre a coleção de itens paginados.

```
// Helper method to work with paginated collection of items directly.  
listRes.contents().stream()  
    .forEach(content -> System.out.println(" Key: " + content.key() + "  
size = " + content.size()));
```

Veja o [exemplo completo](#) em GitHub.

Use um loop para cada

Como `SdkIterable` estende a `Iterable` interface, você pode processar o conteúdo como qualquer `outIterable`. O trecho a seguir usa um `for-each` loop padrão para percorrer o conteúdo da resposta.

```
for (S3Object content : listRes.contents()) {  
    System.out.println(" Key: " + content.key() + " size = " + content.size());  
}
```

Veja o [exemplo completo](#) em GitHub.

Paginação manual

Se seu caso de uso exigir isto, a paginação manual ainda estará disponível. Use o próximo token no objeto de resposta para as solicitações subsequentes. O exemplo a seguir usa um `while` loop.

```
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();
}
```

Veja o [exemplo completo](#) em GitHub.

Paginação assíncrona

Os exemplos a seguir demonstram métodos de paginação assíncrona para listar tabelas. DynamoDB

Itere sobre páginas de nomes de tabelas

Os dois exemplos a seguir usam um cliente assíncrono do DynamoDB que chama `ListTablesPaginator` o método com uma solicitação para obter um. [ListTablesPublisher](#) `ListTablesPublisher` implementa duas interfaces, que oferecem muitas opções para processar respostas. Examinaremos os métodos de cada interface.

Use um **Subscriber**

O exemplo de código a seguir demonstra como processar resultados paginados usando a `org.reactivestreams.Publisher` interface implementada pelo `ListTablesPublisher`. Para saber mais sobre o modelo de fluxos reativos, consulte o repositório [Reactive Streams](#) GitHub.

As importações a seguir se aplicam a todos os exemplos nesta seção de paginação assíncrona.

Importações

```
import io.reactivex.rxjava3.core.Flowable;
import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import reactor.core.publisher.Flux;
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 java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
```

O código a seguir adquire uma `ListTablesPublisher` instância.

```
// Creates a default client with credentials and region loaded from the
environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(listTablesRequest);
```

O código a seguir usa uma implementação anônima de `org.reactivestreams.Subscriber` para processar os resultados de cada página.

O método `onSubscribe` chama o método `Subscription.request` para iniciar solicitações de dados do editor. Esse método deve ser chamado para iniciar a obtenção de dados do editor.

O `onNext` método do assinante processa uma página de resposta acessando todos os nomes das tabelas e imprimindo cada um. Depois que a página é processada, outra página é solicitada ao editor. Esse método é chamado repetidamente até que todas as páginas sejam recuperadas.

O método `onError` será acionado se ocorrer um erro durante a recuperação de dados. Por fim, o método `onComplete` será chamado quando todas as páginas tiverem sido solicitadas.

```
// 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);
    // After 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.
}
});
```

Veja o [exemplo completo](#) em GitHub.

Use um **Consumer**

A `SdkPublisher` interface que `ListTablesPublisher` implementa tem um `subscribe` método que pega a `Consumer` e retorna a `CompletableFuture<Void>`

O `subscribe` método dessa interface pode ser usado para casos de uso simples, quando uma sobrecarga `org.reactivestreams.Subscriber` pode ser demais. Como o código abaixo consome cada página, ele chama o `tableNames` método em cada uma. O `tableNames` método retorna um `java.util.List` dos nomes de tabela do DynamoDB que são processados com o método `forEach`

```
// Use a Consumer for simple use cases.
CompletableFuture<Void> future = publisher.subscribe(
    response -> response.tableNames()
        .forEach(System.out::println));
```

Veja o [exemplo completo](#) em GitHub.

Itere sobre os nomes das tabelas

Os exemplos a seguir mostram maneiras de iterar sobre os objetos retornados na resposta e não nas páginas de resposta. Semelhante ao exemplo síncrono do Amazon S3 mostrado anteriormente com contents seu método, a classe de resultados assíncronos do DynamoDB tem o método conveniente de interagir com `tableNames` a coleção de `ListTablesPublisher` itens subjacente. O tipo de retorno do `tableNames` método é [SdkPublisher](#) aquele que pode ser usado para solicitar itens em todas as páginas.

Use um **Subscriber**

O código a seguir adquire uma `SdkPublisher` parte da coleção subjacente de nomes de tabelas.

```
// Create a default client with credentials and region loaded from the
environment.
final DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();

ListTablesRequest listTablesRequest =
ListTablesRequest.builder().limit(3).build();
ListTablesPublisher listTablesPublisher =
asyncClient.listTablesPaginator(listTablesRequest);
SdkPublisher<String> publisher = listTablesPublisher.tableNames();
```

O código a seguir usa uma implementação anônima de `org.reactivestreams.Subscriber` para processar os resultados de cada página.

O `onNext` método do assinante processa um elemento individual da coleção. Nesse caso, é um nome de tabela. Depois que o nome da tabela é processado, outro nome de tabela é solicitado ao editor. Esse método é chamado repetidamente até que todos os nomes das tabelas sejam recuperados.

```
// Use a 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() {
    }
});
```

Veja o [exemplo completo](#) em GitHub.

Use um **Consumer**

O exemplo a seguir usa o `subscribe` método de `SdkPublisher` que leva a `Consumer` para processar cada item.

```
// Use a Consumer.
CompletableFuture<Void> future = publisher.subscribe(System.out::println);
future.get();
```

Veja o [exemplo completo](#) em GitHub.

Use uma biblioteca de terceiros

Você pode usar outras bibliotecas de terceiros em vez de implementar um assinante personalizado. Este exemplo demonstra o uso de RxJava, mas qualquer biblioteca que implemente as interfaces de fluxo reativo pode ser usada. Consulte a [página RxJava wiki GitHub](#) para obter mais informações sobre essa biblioteca.

Para usar a biblioteca, adicione-a como uma dependência. Se estiver usando o Maven, o exemplo mostra o trecho POM a ser usado.

Entrada POM

```
<dependency>
    <groupId>io.reactivex.rxjava3</groupId>
    <artifactId>rxjava</artifactId>
    <version>3.1.6</version>
</dependency>
```

Código

```
DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create();
ListTablesPublisher publisher =
asyncClient.listTablesPaginator(ListTablesRequest.builder()
    .build());

// The Flowable class has many helper methods that work with
// an implementation of an org.reactivestreams.Publisher.
List<String> tables = Flowable.fromPublisher(publisher)
    .flatMapIterable(ListTablesResponse::tableNames)
    .toList()
    .blockingGet();
System.out.println(tables);
```

AWSCliente S3 baseado em CRT

O cliente S3 AWS baseado em CRT, construído com base no [AWSCommon Runtime \(CRT\)](#), é um [cliente assíncrono](#) S3 alternativo. [Ele transfere objetos de e para o Amazon Simple Storage Service \(Amazon S3\)](#) com desempenho e confiabilidade aprimorados usando automaticamente a API de upload de várias partes e as buscas de intervalo de bytes do Amazon S3.

O cliente S3 AWS baseado em CRT melhora a confiabilidade da transferência caso haja uma falha na rede. A confiabilidade é aprimorada ao tentar novamente partes individuais com falha de uma transferência de arquivo sem reiniciar a transferência desde o início.

Além disso, o cliente S3 AWS baseado em CRT oferece pool de conexões aprimorado e balanceamento de carga do Sistema de Nomes de Domínio (DNS), o que também melhora a taxa de transferência.

Você pode usar o cliente S3 AWS baseado em CRT no lugar do cliente assíncrono S3 padrão do SDK e aproveitar imediatamente sua taxa de transferência aprimorada.

AWSComponentes baseados em CRT no SDK

O cliente S3 AWS baseado em CRT, descrito neste tópico, e o cliente HTTP AWS baseado em CRT são componentes diferentes no SDK.

O cliente S3 AWS baseado em CRT é uma implementação da `AsyncClient` interface [S3](#) e é usado para trabalhar com o serviço Amazon S3. É uma alternativa à implementação da `S3AsyncClient` interface baseada em Java e oferece vários benefícios.

O [cliente HTTP AWS baseado em CRT](#) é uma implementação da `SdkAsyncHttpClient` interface e é usado para comunicação HTTP geral. É uma alternativa à implementação Netty da `SdkAsyncHttpClient` interface e oferece várias vantagens.

Embora ambos os componentes usem bibliotecas do [AWSCommon Runtime](#), o cliente S3 AWS baseado em CRT usa a [biblioteca aws-c-s 3 e oferece suporte aos recursos](#) da API de upload de [várias partes do S3](#). Como o cliente HTTP AWS baseado em CRT é destinado ao uso geral, ele não oferece suporte aos recursos da API de upload de várias partes do S3.

Adicione dependências para usar o cliente S3 baseado em AWS CRT

Para usar o cliente S3 AWS baseado em CRT, adicione as duas dependências a seguir ao seu arquivo de projeto Maven. O exemplo mostra as versões mínimas a serem usadas. [Pesquise no repositório central do Maven as versões mais recentes dos artefatos s3 e aws-crt](#).

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
    <version>2.20.68</version>
</dependency>
<dependency>
    <groupId>software.amazon.awssdk.crt</groupId>
    <artifactId>aws-crt</artifactId>
    <version>0.21.16</version>
</dependency>
```

Crie uma instância do cliente S3 AWS baseado em CRT

Crie uma instância do cliente S3 AWS baseado em CRT com as configurações padrão, conforme mostrado no trecho de código a seguir.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

Para configurar o cliente, use o AWS CRT Client Builder. Você pode alternar do cliente assíncrono S3 padrão para o cliente AWS baseado em CRT alterando o método builder.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;

S3AsyncClient s3AsyncClient =
    S3AsyncClient.crtBuilder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_WEST_2)
        .targetThroughputInGbps(20.0)
        .minimumPartSizeInBytes(8 * 1025 * 1024L)
        .build();
```

 Note

Algumas das configurações no construtor padrão podem não ser suportadas atualmente no construtor de clientes AWS CRT. Ligue para o construtor padrão `S3AsyncClient#builder()`.

Use o cliente AWS S3 baseado em CRT

Use o cliente S3 AWS baseado em CRT para chamar as operações de API do Amazon S3. O exemplo a seguir demonstra as [GetObject](#) operações [PutObject](#) disponíveis por meio do AWS SDK for Java.

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.AsyncResponseTransformer;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

S3AsyncClient s3Client = S3AsyncClient.crtCreate();

// Upload a local file to Amazon S3.
PutObjectResponse putObjectResponse =
    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 getObjectResponse =
    s3Client.getObject(req -> req.bucket(<BUCKET_NAME>)
        .key(<KEY_NAME>),
        AsyncResponseTransformer.toFile(Paths.get(<FILE_NAME>)))
    .join();
```

Gerenciador de transferências do Amazon S3

O Amazon S3 Transfer Manager é um utilitário de transferência de arquivos de alto nível e de código aberto para o AWS SDK for Java 2.x Use-o para transferir arquivos e diretórios de e para o Amazon Simple Storage Service (Amazon S3).

Quando construído sobre [the section called “AWSClient S3 baseado em CRT”](#), o S3 Transfer Manager pode aproveitar as melhorias de desempenho, como [API de upload de várias partes](#) e [buscas por intervalo de bytes](#).

Com o S3 Transfer Manager, você também pode monitorar o progresso de uma transferência em tempo real e pausar a transferência para execução posterior.

Conceitos básicos

Adicione dependências ao seu arquivo de compilação

Para usar o S3 Transfer Manager com desempenho aprimorado com base no cliente S3 AWS baseado em CRT, configure seu arquivo de compilação com as seguintes dependências.

- Use a versão [2.19.1](#) ou superior do SDK for Java 2.x.
- Adicione o `s3-transfer-manager` artefato como uma dependência.
- Adicione o `aws-crt` artefato como uma dependência na versão [0.20.3](#) ou superior.

O exemplo de código a seguir mostra como configurar as dependências do seu projeto para o 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>
```

[Pesquise no repositório central do Maven as versões mais recentes dos artefatos s3-transfer-manager e aws-crt.](#)

Crie uma instância do S3 Transfer Manager

O trecho a seguir mostra como criar uma TransferManager instância do [S3](#) com configurações padrão.

```
S3TransferManager transferManager = S3TransferManager.create();
```

O exemplo a seguir mostra como configurar um S3 Transfer Manager com configurações personalizadas. Neste exemplo, uma AsyncClient instância [S3 AWS baseada em CRT](#) é usada como cliente subjacente para o 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

Se a aws-crt dependência não estiver incluída no arquivo de compilação, o S3 Transfer Manager será criado com base no cliente assíncrono S3 padrão usado no SDK for Java 2.x.

Carregar um arquivo em um bucket do S3

[Para fazer upload de um arquivo para o Amazon S3 usando o S3 Transfer Manager, passe um UploadFileRequest objeto para o método uploadFile S3TransferManager do Amazon.](#)

O [FileUpload](#) objeto retornado do `uploadFile` método representa o processo de upload. Depois que a solicitação for concluída, o [CompletedFileUpload](#) objeto conterá informações sobre o upload.

O exemplo a seguir mostra um exemplo de upload de arquivo junto com o uso opcional de um [LoggingTransferListener](#), que registra o progresso do upload.

Importações

```
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;
```

Código

```
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();
}
```

Baixar um arquivo de um bucket do S3

Para baixar um objeto de um bucket do S3 usando o S3 Transfer Manager, crie um [DownloadFileRequest](#) objeto e passe-o para o método [downloadFile](#).

O [FileDownload](#) objeto retornado pelo [downloadFile](#) método [S3TransferManager](#)'s representa a transferência do arquivo. Após a conclusão do download, ele [CompletedFileDownload](#) contém acesso às informações sobre o download.

O exemplo a seguir também mostra um exemplo de download mais o uso opcional de um [LoggingTransferListener](#), que registra o progresso do download.

Importações

```
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;
```

Código

```
public Long downloadFile(S3TransferManager transferManager, String bucketName,
                         String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest =
        DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .addTransferListener(LoggingTransferListener.create())
            .destination(Paths.get(downloadedFilePath))
            .build();

    FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
    logger.info("Content length [{}]", downloadResult.response().contentLength());
    return downloadResult.response().contentLength();
}
```

Copiar um objeto do Amazon S3 para outro bucket

Para começar a cópia de um objeto de um bucket do S3 para outro bucket, crie uma [CopyObjectRequest](#)instância básica.

Em seguida, envolva o básico `CopyObjectRequest` em um [CopyRequest](#)que possa ser usado pelo S3 Transfer Manager.

O Copy objeto retornado pelo copy método `S3TransferManager's` representa o processo de cópia. Depois que o processo de cópia for concluído, o [CompletedCopy](#)objeto conterá detalhes sobre a resposta.

Importações

```
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;
```

Código

```
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

Atualmente, não há suporte para cópias entre regiões.

Carregar um diretório local em um bucket do S3

Para fazer upload de um diretório local para um bucket do S3, comece chamando o método [uploadDirectory](#) da `S3TransferManager` instância, passando um. [UploadDirectoryRequest](#)

O `DirectoryUpload` objeto representa o processo de upload, que gera um [CompletedDirectoryUpload](#) quando a solicitação é concluída. O `CompleteDirectoryUpload` objeto

contém informações sobre os resultados da transferência, incluindo quais arquivos falharam na transferência.

Importações

```
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;
```

Código

```
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();
}
```

Baixe objetos de bucket do S3 para um diretório local

Para baixar os objetos em um bucket do S3 para um diretório local, comece chamando o método [DownloadDirectory](#) do Transfer Manager, passando um. [DownloadDirectoryRequest](#)

O [DirectoryDownload](#) objeto representa o processo de download, que gera um [CompletedDirectoryDownload](#) quando a solicitação é concluída. O [CompleteDirectoryDownload](#)

objeto contém informações sobre os resultados da transferência, incluindo quais arquivos falharam na transferência.

Importações

```
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;
```

Código

```
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();
}
```

Use garçons no 2.x AWS SDK for Java

O utilitário waiters do AWS SDK for Java 2.x permite que você valide se AWS os recursos estão em um estado especificado antes de realizar operações nesses recursos.

Um garçom é uma abstração usada para pesquisar AWS recursos, como DynamoDB tabelas ou Amazon S3 compartimentos, até que o estado desejado seja alcançado (ou até que seja determinado que o recurso nunca alcançará o estado desejado). Em vez de escrever uma lógica para pesquisar continuamente seus AWS recursos, o que pode ser complicado e propenso a erros, você pode usar garçons para pesquisar um recurso e fazer com que seu código continue sendo executado depois que o recurso estiver pronto.

Pré-requisitos

Antes de usar garçons em um projeto com o AWS SDK for Java, você deve concluir as etapas em [Configurando](#) o 2.x. AWS SDK for Java

Você também deve configurar as dependências do seu projeto (por exemplo, no seu `build.gradle` arquivo `pom.xml` ou) para usar a versão 2.15.0 ou posterior do AWS SDK for Java.

Por exemplo:

```
<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>
```

Usando garçons

Para instanciar um objeto de garçons, primeiro crie um cliente de serviço. Defina o `waiter()` método do cliente de serviço como o valor do objeto garçom. Quando a instância do waiter existir, defina suas opções de resposta para executar o código apropriado.

Programação síncrona

O trecho de código a seguir mostra como esperar que uma DynamoDB tabela exista e esteja em um estado ATIVO.

```
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);
```

Programação assíncrona

O trecho de código a seguir mostra como esperar que uma DynamoDB tabela não exista mais.

```
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();
```

Configurar garçons

Você pode personalizar a configuração de um garçom usando o `overrideConfiguration()` em seu construtor. Para algumas operações, você pode aplicar uma configuração personalizada ao fazer a solicitação.

Configurar um garçom

O trecho de código a seguir mostra como substituir a configuração em um garçom.

```
// 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();
```

Substituir configuração para uma solicitação específica

O trecho de código a seguir mostra como substituir a configuração de um garçom por solicitação. Observe que somente algumas operações têm configurações personalizáveis.

```
waiter.waitUntilTableNotExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
    o -> o.waitTimeout(Duration.ofMinutes(1)));
```

Exemplos de código

Para ver um exemplo completo do uso de garçons com DynamoDB, consulte [CreateTable.java](#) no Repositório de exemplos de AWS código.

Para ver um exemplo completo do uso de garçons com Amazon S3, consulte [S3 BucketOps.java](#) no Repositório de exemplos de código. AWS

API IAM Policy Builder

A [API IAM Policy Builder](#) é uma biblioteca que você pode usar para criar [políticas do IAM](#) em Java e carregá-las no AWS Identity and Access Management (IAM).

Em vez de criar uma política do IAM montando manualmente uma string JSON ou lendo um arquivo, a API fornece uma abordagem orientada a objetos do lado do cliente para gerar a string

JSON. Quando você lê uma política do IAM existente no formato JSON, a API a converte em uma [IamPolicy](#) instância para manipulação.

A API IAM Policy Builder ficou disponível com a versão 2.20.105 do SDK, então use essa versão ou uma versão posterior em seu arquivo de compilação do Maven. O número da versão mais recente do SDK está [listado na central do Maven](#).

O trecho a seguir mostra um exemplo de bloco de dependência para um arquivo Maven. pom.xml. Isso permite que você use a API IAM Policy Builder em seu projeto.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam-policy-builder</artifactId>
    <version>2.20.139</version>
</dependency>
```

Criar um IamPolicy.

Esta seção mostra vários exemplos de como criar políticas usando a API IAM Policy Builder.

Em cada um dos exemplos a seguir, comece com o [IamPolicy.Builder](#) e adicione uma ou mais declarações usando o addStatement método. Seguindo esse padrão, o [IamStatement.Builder](#) tem métodos para adicionar o efeito, as ações, os recursos e as condições à declaração.

Exemplo: criar uma política baseada em tempo

O exemplo a seguir cria uma política baseada em identidade que permite a ação do Amazon DynamoDB entre dois momentos. GetItem

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.DATE_GREATER_THAN)
                .key("aws:CurrentTime")
                .value("2020-04-01T00:00:00Z"))
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.DATE_LESS_THAN))
```

```
        .key("aws:CurrentTime")
        .value("2020-06-30T23:59:59Z")))
    .build();

    // Use an IamPolicyWriter to write out the JSON string to a more readable
format.
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}
```

Saída JSON

A última declaração no exemplo anterior retorna a seguinte string JSON.

Leia mais sobre esse [exemplo](#) no Guia do AWS Identity and Access Management usuário.

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Effect" : "Allow",
            "Action" : "dynamodb:GetItem",
            "Resource" : "*",
            "Condition" : {
                "DateGreaterThan" : {
                    "aws:CurrentTime" : "2020-04-01T00:00:00Z"
                },
                "DateLessThan" : {
                    "aws:CurrentTime" : "2020-06-30T23:59:59Z"
                }
            }
        }
    ]
}
```

Exemplo: especifique várias condições

O exemplo a seguir mostra como você pode criar uma política baseada em identidade que permita o acesso a atributos específicos do DynamoDB. A política contém duas condições.

```
public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
```

```

        .addAction("dynamodb:GetItem")
        .addAction("dynamodb:BatchGetItem")
        .addAction("dynamodb:Query")
        .addAction("dynamodb:PutItem")
        .addAction("dynamodb:UpdateItem")
        .addAction("dynamodb:DeleteItem")
        .addAction("dynamodb:BatchWriteItem")
        .addResource("arn:aws:dynamodb:*:*:table/table-name")

.addConditions(IamConditionOperator.STRING_EQUALS.addPrefix("ForAllValues:"),  

                "dynamodb:Attributes",
                List.of("column-name1", "column-name2", "column-  

name3))
            .addCondition(b1 ->  

b1.operator(IamConditionOperator.STRING_EQUALS.addSuffix("IfExists"))
            .key("dynamodb:Select")
            .value("SPECIFIC_ATTRIBUTES")))
        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Saída JSON

A última declaração no exemplo anterior retorna a seguinte string JSON.

Leia mais sobre esse [exemplo](#) no Guia do AWS Identity and Access Management usuário.

```
{
    "Version" : "2012-10-17",
    "Statement" : {
        "Effect" : "Allow",
        "Action" : [ "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",
        "dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem" ],
        "Resource" : "arn:aws:dynamodb:*:*:table/table-name",
        "Condition" : {
            "ForAllValues:StringEquals" : {
                "dynamodb:Attributes" : [ "column-name1", "column-name2", "column-name3" ]
            },
            "StringEqualsIfExists" : {
                "dynamodb:Select" : "SPECIFIC_ATTRIBUTES"
            }
        }
    }
}
```

```
    }  
}
```

Exemplo: especifique os principais

O exemplo a seguir mostra como criar uma política baseada em recursos que nega acesso a um bucket para todos os principais, exceto aqueles especificados na condição.

```
public String specifyPrincipalsExample() {  
    IamPolicy policy = IamPolicy.builder()  
        .addStatement(b -> b  
            .effect(IamEffect.DENY)  
            .addAction("s3:*")  
            .addPrincipal(IamPrincipal.ALL)  
            .addResource("arn:aws:s3::::BUCKETNAME/*")  
            .addResource("arn:aws:s3::::BUCKETNAME")  
            .addCondition(b1 -> b1  
                .operator(IamConditionOperator.ARN_NOT_EQUALS)  
                .key("aws:PrincipalArn")  
                .value("arn:aws:iam::444455556666:user/user-name")))  
        .build();  
    return policy.toJson(IamPolicyWriter.builder()  
        .prettyPrint(true).build());  
}
```

Saída JSON

A última declaração no exemplo anterior retorna a seguinte string JSON.

Leia mais sobre esse [exemplo](#) no Guia do AWS Identity and Access Management usuário.

```
{  
    "Version" : "2012-10-17",  
    "Statement" : {  
        "Effect" : "Deny",  
        "Principal" : "*",  
        "Action" : "s3:*",  
        "Resource" : [ "arn:aws:s3::::BUCKETNAME/*", "arn:aws:s3::::BUCKETNAME" ],  
        "Condition" : {  
            "ArnNotEquals" : {  
                "aws:PrincipalArn" : "arn:aws:iam::444455556666:user/user-name"  
            }  
        }  
    }  
}
```

```
    }  
}  
}
```

Exemplo: permitir acesso entre contas

O exemplo a seguir mostra como permitir que outra pessoa Conta da AWS faça upload de objetos para o seu bucket, mantendo o controle total do proprietário dos objetos enviados.

```
public String allowCrossAccountAccessExample() {  
    IamPolicy policy = IamPolicy.builder()  
        .addStatement(b -> b  
            .effect(IamEffect.ALLOW)  
            .addPrincipal(IamPrincipalType.AWS, "111122223333")  
            .addAction("s3:PutObject")  
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")  
            .addCondition(b1 -> b1  
                .operator(IamConditionOperator.STRING_EQUALS)  
                .key("s3:x-amz-acl")  
                .value("bucket-owner-full-control")))  
        .build();  
    return policy.toJson(IamPolicyWriter.builder()  
        .prettyPrint(true).build());  
}
```

Saída JSON

A última declaração no exemplo anterior retorna a seguinte string JSON.

Leia mais sobre esse [exemplo](#) no Guia do usuário do Amazon Simple Storage Service.

```
{  
    "Version" : "2012-10-17",  
    "Statement" : {  
        "Effect" : "Allow",  
        "Principal" : {  
            "AWS" : "111122223333"  
        },  
        "Action" : "s3:PutObject",  
        "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",  
        "Condition" : {  
            "StringEquals" : {
```

```
        "s3:x-amz-acl" : "bucket-owner-full-control"
    }
}
}
}
```

Use e IamPolicy com IAM

Depois de criar uma `IamPolicy` instância, você usa um [IamClient](#) para trabalhar com o serviço IAM.

O exemplo a seguir cria uma política que permite que uma [identidade do IAM](#) grave itens em uma tabela do DynamoDB na conta especificada com o parâmetro `accountID`. Em seguida, a política é enviada para o IAM como uma string JSON.

```
public String createAndUploadPolicyExample(IamClient iam, String accountID, String
policyName) {
    // Build the policy.
    IamPolicy policy =
        IamPolicy.builder() // 'version' defaults to "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-east-1:" + accountID
+ ":table/exampleTableName")
            .build())
            .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}
```

O próximo exemplo se baseia no exemplo anterior. O código baixa a política e a usa como base para uma nova política, copiando e alterando a declaração. A nova política é então carregada.

```
public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName, String newPolicyName) {

    String policyArn = "arn:aws:iam::" + accountID + ":policy/" + policyName;
    GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));
```

```
String policyVersion = getPolicyResponse.policy().defaultVersionId();
GetPolicyVersionResponse getPolicyVersionResponse =
    iam.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

    // Create an IamPolicy instance from the JSON string returned from IAM.
    String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
    IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

    /*
        All IamPolicy components are immutable, so use the copy method that
creates a new instance that
        can be altered in the same method call.

        Add the ability to get an item from DynamoDB as an additional action.
    */
    IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

    // Create a new statement that replaces the original statement.
    IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

    // Upload the new policy. IAM now has both policies.
    iam.createPolicy(r -> r.policyName(newPolicyName)
        .policyDocument(newPolicy.toJson()));

    return newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}
```

IamClient

Os exemplos anteriores usam um `IamClient` argumento criado conforme mostrado no trecho a seguir.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

Políticas em JSON

Os exemplos retornam as seguintes cadeias de caracteres JSON.

```
First example
{
    "Version" : "2012-10-17",
    "Statement" : {
        "Effect" : "Allow",
        "Action" : "dynamodb:PutItem",
        "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
    }
}
```

Second example

```
{
    "Version" : "2012-10-17",
    "Statement" : {
        "Effect" : "Allow",
        "Action" : [ "dynamodb:PutItem", "dynamodb:GetItem" ],
        "Resource" : "arn:aws:dynamodb:us-east-1:111122223333:table/exampleTableName"
    }
}
```

Trabalhe com Serviços da AWS o uso do AWS SDK for Java 2.x

Esta seção fornece tutoriais curtos e orientações sobre como trabalhar com. Serviços da AWS

Tópicos

- [Trabalhar com CloudWatch](#)
- [Trabalhar com Amazon Cognito](#)
- [AWSserviços de banco de dados e AWS SDK for Java 2.x](#)
- [Trabalhar com DynamoDB](#)
- [Trabalhar com Amazon EC2](#)
- [Trabalhar com IAM](#)
- [Trabalhar com Kinesis](#)
- [Invocar, listar e excluir funções AWS Lambda](#)
- [Trabalhar com Amazon Pinpoint](#)
- [Trabalhe com o Amazon S3](#)
- [Trabalhar com Amazon Simple Notification Service](#)
- [Trabalhar com Amazon Simple Queue Service](#)
- [Trabalhar com Amazon Transcribe](#)

Trabalhar com CloudWatch

Esta seção fornece exemplos de programação [CloudWatch](#) usando o AWS SDK for Java 2.x.

O Amazon CloudWatch monitora seus recursos da Amazon Web Services (AWS) e os aplicativos que você executa na AWS em tempo real. Você pode usar o CloudWatch para coletar e monitorar métricas, que são as variáveis que podem ser medidas para avaliar seus recursos e aplicativos. Os alarmes do CloudWatch enviam notificações ou fazem alterações automaticamente nos recursos que você está monitorando com base nas regras definidas.

Para obter mais informações sobre o CloudWatch, consulte o [Guia do usuário do Amazon CloudWatch](#).

Os exemplos a seguir incluem somente o código necessário para demonstrar cada técnica. O [código de exemplo completo está disponível em GitHub](#). A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

- [Obtenha métricas de CloudWatch](#)
- [Publique dados métricos personalizados em CloudWatch](#)
- [Trabalhe com CloudWatch alarms](#)
- [Use ações de alarme em CloudWatch](#)
- [Enviar eventos para CloudWatch](#)

Obtenha métricas de CloudWatch

Listar métricas

Para listar CloudWatch métricas, crie um [ListMetricsRequest](#) e chame o `listMetrics` método `CloudWatchClient`'s. Você pode usar o `ListMetricsRequest` para filtrar as métricas retornadas por namespace, nome da métrica ou dimensões.

Note

Uma lista de métricas e dimensões publicadas pelos AWS serviços pode ser encontrada na [Referência de Amazon CloudWatch métricas e dimensões](#) no Guia do Amazon CloudWatch usuário.

Importações

```
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;
```

Código

```
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);  
    }  
}
```

As métricas são retornadas em um [ListMetricsResponse](#) chamando seu `getMetrics` método.

Os resultados podem ser paginados. Para recuperar o próximo lote de resultados, chame `nextToken` no objeto de resposta e use o valor do token para compilar um novo objeto de solicitação. Em seguida, chame o método `listMetrics` novamente com a nova solicitação.

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [ListMetrics](#) na referência da Amazon CloudWatch API

Publique dados métricos personalizados em CloudWatch

Vários AWS serviços publicam [suas próprias métricas](#) em namespaces que começam com "AWS". Você também pode publicar dados métricos personalizados usando seu próprio namespace (desde que não comece com ""). AWS

Publicar dados de métrica personalizada

Para publicar seus próprios dados métricos, chame o `putMetricData` método `CloudWatchClient's` com um [PutMetricDataRequest](#). Eles `PutMetricDataRequest` devem incluir o namespace personalizado a ser usado para os dados e informações sobre o próprio ponto de dados em um [MetricDatum](#) objeto.

 Note

Você não pode especificar um namespace que comece com "". AWS Os namespaces que começam com "AWS" são reservados para uso por Amazon Web Services produtos.

Importações

```
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;
```

Código

```
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);
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [Usando Amazon CloudWatch métricas](#) no Guia do Amazon CloudWatch usuário.
- [AWSNamespaces](#) no Guia do Amazon CloudWatch usuário.
- [PutMetricData](#) na referência da Amazon CloudWatch API.

Trabalhe com CloudWatch alarms

Criar um alarme

Para criar um alarme com base em uma CloudWatch métrica, chame o `putMetricAlarm` método `CloudWatchClient's PutMetricAlarmRequest` preenchido com as condições do alarme.

Importações

```
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;
```

Código

```
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);
}
}
```

Veja o [exemplo completo](#) em GitHub.

Listar alarmes

Para listar os CloudWatch alarmes que você criou, chame o `describeAlarms` método `CloudWatchClient's` com um [DescribeAlarmsRequest](#) que você possa usar para definir opções para o resultado.

Importações

```
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;
```

Código

```
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");
}
```

A lista de alarmes pode ser obtida `MetricAlarms` chamando o [DescribeAlarmsResponse](#) que é retornado por `describeAlarms`.

Os resultados podem ser paginados. Para recuperar o próximo lote de resultados, chame `nextToken` no objeto de resposta e use o valor do token para compilar um novo objeto de solicitação. Em seguida, chame o método `describeAlarms` novamente com a nova solicitação.

Note

Você também pode recuperar alarmes para uma métrica específica usando o método `describeAlarmsForMetric` do `CloudWatchClient`. O uso é semelhante a `describeAlarms`.

Veja o [exemplo completo](#) emGitHub.

Excluir alarmes

Para excluir CloudWatch alarmes, chame o `deleteAlarms` método `CloudWatchClient`'s [`DeleteAlarmsRequest`](#) contendo um ou mais nomes de alarmes que você deseja excluir.

Importações

```
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;
```

Código

```
public static void deleteCWAAlarm(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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [Criação Amazon CloudWatch de alarmes](#) no Guia do Amazon CloudWatch usuário
- [PutMetricAlarm](#)na referência da Amazon CloudWatch API
- [DescribeAlarms](#)na referência da Amazon CloudWatch API
- [DeleteAlarms](#)na referência da Amazon CloudWatch API

Use ações de alarme em CloudWatch

Usando ações de alarme do CloudWatch, é possível criar alarmes que realizam ações como interromper, encerrar, reiniciar ou recuperar automaticamente instâncias do Amazon EC2.

Note

As ações de alarme podem ser adicionadas a um alarme usando o `alarmActions` método [PutMetricAlarmRequest](#)'s ao [criar um alarme](#).

Habilitar ações de alarme

Para habilitar ações de CloudWatch alarme para um alarme, chame os `CloudWatchClient` [EnableAlarmActionsRequest](#) contendo um ou mais nomes de alarmes cujas ações você deseja ativar. `enableAlarmActions`

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Desabilitar ações de alarme

Para desativar as ações de CloudWatch alarme de um alarme, chame os `CloudWatchClient` [`DisableAlarmActionsRequest`](#) contendo um ou mais nomes de alarmes cujas ações você deseja desativar. `disableAlarmActions`

Importações

```
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;
```

Código

```
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);
    }
}
```

}

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [Crie alarmes para interromper, encerrar, reiniciar ou recuperar uma instância](#) no Guia do usuário Amazon CloudWatch
- [PutMetricAlarm](#)na referência da Amazon CloudWatch API
- [EnableAlarmActions](#)na referência da Amazon CloudWatch API
- [DisableAlarmActions](#)na referência da Amazon CloudWatch API

Enviar eventos para CloudWatch

CloudWatch Eventsfornece um fluxo quase em tempo real de eventos do sistema que descrevem mudanças nos AWS recursos em Amazon EC2 instâncias, Lambda funções, Kinesis fluxos, Amazon ECS tarefas, máquinas de Step Functions estado, Amazon SNS tópicos, Amazon SQS filas ou alvos integrados. Você pode comparar eventos e roteá-los para um ou mais fluxos ou funções de destino usando regras simples.

Adicionar eventos

Para adicionar CloudWatch eventos personalizados, chame o `putEvents` método `CloudWatchEventsClient's` com um [PutEventsRequest](#)objeto que contém um ou mais [PutEventsRequestEntry](#)objetos que fornecem detalhes sobre cada evento. Você pode especificar vários parâmetros para a entrada, como a origem e o tipo do evento, recursos associados ao evento e assim por diante.

 Note

Você pode especificar um máximo de dez eventos por chamada para `putEvents`.

Importações

```
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;
```

Código

```
public static void putCWEEvents(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);  
    }  
}
```

Veja o [exemplo completo](#) em GitHub.

Adicionar regras

Para criar ou atualizar uma regra, chame o `putRule` método `CloudWatchEventsClient's` com a [PutRuleRequest](#) com o nome da regra e parâmetros opcionais, como o [padrão do evento](#), a IAM função a ser associada à regra e uma [expressão de agendamento](#) que descreva a frequência com que a regra é executada.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Adicionar destinos

Destinos são os recursos invocados quando uma regra é disparada. Entre os destinos de exemplo estão instâncias do Amazon EC2, funções do Lambda, streamings do Kinesis, tarefas do Amazon ECS, máquinas de estado do Step Functions e destinos integrados.

Para adicionar um alvo a uma regra, chame o `putTargets` método `CloudWatchEventsClient's PutTargetsRequest` contendo a regra a ser atualizada e uma lista de destinos a serem adicionados à regra.

Importações

```
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;
```

Código

```
public static void putCWTTargets(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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [Adicionar eventos com PutEvents](#) no Guia do Amazon CloudWatch Events usuário
- [Agende expressões para regras](#) no Guia Amazon CloudWatch Events do usuário
- [Tipos de eventos para CloudWatch Events](#) no Guia Amazon CloudWatch Events do usuário
- [Eventos e padrões de eventos](#) no Guia Amazon CloudWatch Events do usuário
- [PutEvents](#)na referência da Amazon CloudWatch Events API

- [PutTargets](#)na referência da Amazon CloudWatch Events API
- [PutRule](#)na referência da Amazon CloudWatch Events API

Trabalhar com Amazon Cognito

Com o Amazon Cognito, é possível adicionar rapidamente o recurso de cadastro ou login do usuário ao aplicativo web ou móvel. Os exemplos aqui demonstram algumas das funcionalidades básicas doAmazon Cognito.

Criar um grupo de usuários

Um grupo de usuários é um diretório de usuários que você pode configurar para o aplicativo web ou móvel.

Para criar um grupo de usuários, comece construindo um [CreateUserPoolRequest](#)objeto, com o nome do grupo de usuários como o valor do `userPoolName()`. Chame o método `createUserPool()` do [CreateUserPoolRequest](#) transmitindo o objeto `CreateUserPoolRequest`. Você pode capturar o resultado dessa solicitação como um [CreateUserPoolResponse](#)objeto, conforme demonstrado no trecho de código a seguir.

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) emGitHub.

Listar usuários de um grupo de usuários

Para listar usuários de seus grupos de usuários, comece criando um [ListUserPoolsRequest](#) objeto, com o número máximo de resultados como o valor de seu `maxResults()`. Chame o método `listUserPools()` do `CognitoIdentityProviderClient` transmitindo o objeto `ListUserPoolsRequest`. Você pode capturar o resultado dessa solicitação como um [ListUserPoolsResponse](#) objeto, conforme demonstrado no trecho de código a seguir. Crie um [UserPoolDescriptionType](#) objeto para repetir facilmente os resultados e extrair os atributos de cada usuário.

Importações

```
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;
```

Código

```
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);
}
}
```

Veja o [exemplo completo](#) emGitHub.

Criar um grupo de identidades do

Um grupo de identidades é um contêiner que organiza os IDs do provedor de identidade externo, mantendo um identificador exclusivo para cada usuário. Para criar um grupo de identidades, comece criando um [CreateIdentityPoolRequest](#) com o nome do grupo de usuários como o valor de `seuidentityPoolName()`. Defina `allowUnauthenticatedIdentities()` como true ou false. Chame o método `createIdentityPool()` do objeto `CognitoIdentityClient` transmitindo o objeto `CreateIdentityPoolRequest`. Você pode capturar o resultado dessa solicitação como um [CreateIdentityPoolResponse](#) objeto, conforme demonstrado no trecho de código a seguir.

Importações

```
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
```

Código

```
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 "";  
}
```

Veja o [exemplo completo](#) em GitHub.

Adicionar um cliente de aplicativo

Para habilitar a interface de usuário da web hospedada de cadastro ou login para o aplicativo, crie um cliente de aplicativo. Para criar um cliente de aplicativo, comece criando um [CreateUserPoolClientRequest](#) objeto, com o nome do cliente como o valor de `clientName()`.

Defina `userPoolId()` como o ID do grupo de usuários ao qual você deseja anexar esse cliente de aplicativo. Chame o método `createUserPoolClient()` do `CognitoIdentityProviderClient` transmitindo o objeto `CreateUserPoolClientRequest`. Você pode capturar o resultado dessa solicitação como um [CreateUserPoolClientResponse](#) objeto, conforme demonstrado no trecho de código a seguir.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Adicionar um provedor de identidade de terceiros

Adicionar um provedor de identidade externo (IdP) permite que seus usuários façam login no aplicativo usando o mecanismo de login desse serviço. Para adicionar um IdP de terceiros, comece criando um [UpdateIdentityPoolRequest](#) objeto, com o nome do grupo de identidades como o valor do `seuidentityPoolName()`. Defina `allowUnauthenticatedIdentities()` como `true` ou `false`, especifique o `identityPoolId()` e defina quais provedores de login serão compatíveis com `supportedLoginProviders()`. Chame o método `updateIdentityPool()` do `CognitoIdentityClient` transmitindo o objeto `UpdateIdentityPoolRequest`. Você pode

capturar o resultado dessa solicitação como um [UpdateIdentityPoolResponse](#) objeto, conforme demonstrado no trecho de código a seguir.

Importações

```
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;
```

Código

```
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);
        }
    }
```

Veja o [exemplo completo](#) em GitHub.

Obter credenciais para um ID

Para obter as credenciais de uma identidade em um pool de identidades, primeiro crie um [GetCredentialsForIdentityRequest](#) com o ID de identidade como o valor de seu `identityId()`. Chame o método `getCredentialsForIdentity()` do `CognitoIdentityClient` transmitindo o objeto `GetCredentialsForIdentityRequest`. Você pode capturar o resultado dessa solicitação como um [GetCredentialsForIdentityResponse](#) objeto, conforme demonstrado no trecho de código a seguir.

Importações

```
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
```

Código

```
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);  
    }  
}
```

Veja o [exemplo completo](#) em GitHub.

Para obter mais informações, consulte o [Guia do desenvolvedor do Amazon Cognito](#).

AWSserviços de banco de dados e AWS SDK for Java 2.x

AWS [oferece vários tipos de banco de dados: relacional, valor-chave, em memória, documento e vários outros](#). O suporte do SDK for Java 2.x varia de acordo com a natureza do serviço de banco de dados em AWS

Alguns serviços de banco de dados, por exemplo, o serviço [Amazon DynamoDB](#), têm APIs de serviços web para gerenciar o AWS recurso (banco de dados), bem como APIs de serviços web para interagir com os dados. [No SDK for Java 2.x, esses tipos de serviços têm clientes de serviço dedicados, por exemplo, DynamoDBClient](#).

Outros serviços de banco de dados têm APIs de serviços web que interagem com o recurso, como a API [Amazon DocumentDB](#) (para gerenciamento de clusters, instâncias e recursos), mas não têm uma API de serviço web para trabalhar com os dados. O SDK for Java 2.x tem uma [DocDbClient](#) interface correspondente para trabalhar com o recurso. No entanto, você precisa de outra API Java, como o [MongoDB para que o Java](#) funcione com os dados.

Use os exemplos abaixo para saber como você usa o SDK para clientes de serviço Java 2.x com os diferentes tipos de bancos de dados.

Exemplos do Amazon DynamoDB

Trabalhar com dados

SDK service client: [Cliente DynamoDB](#)

Trabalhar com a base de dados

SDK service client: [Cliente DynamoDB](#)

Trabalhar com dados

Example: [Aplicativo REST React/Spring usando o DynamoDB](#)

Examples: [Vários exemplos do DynamoDB](#)

SDK service client: [DynamoDB EnhancedClient](#)

Example: [Aplicativo REST React/Spring usando o DynamoDB](#)

Examples: [Vários exemplos do DynamoDB](#)
(names starting with 'Enhanced')

Trabalhar com a base de dados

Examples: [CreateTable](#), [ListTables](#), [DeleteTable](#)

Veja [outros exemplos do DynamoDB](#) na seção de exemplos de código guiado deste guia.

Exemplos do Amazon RDS

Trabalhar com dados

API não SDK: JDBC, versão SQL específica do banco de dados; seu código gerencia conexões de banco de dados ou um pool de conexões.

Exemplo: aplicativo [React/Spring REST usando MySQL](#)

Trabalhar com a base de dados

Cliente de serviço SDK: [RdsClient](#)

Exemplos: [Vários RdsClient exemplos](#)

Exemplos do Amazon Redshift

Trabalhar com dados

Cliente de serviço SDK: [RedshiftDataClient](#)

Exemplos: [Vários RedshiftDataClient exemplos](#)

Trabalhar com a base de dados

Cliente de serviço SDK: [RedshiftClient](#)

Exemplos: [Vários RedshiftClient exemplos](#)

Trabalhar com dados	Trabalhar com a base de dados
Exemplo: aplicativo React/Spring REST usando RedshiftDataClient	

Exemplos do Amazon Aurora Sem Servidor v1

Trabalhar com dados	Trabalhar com a base de dados
Cliente de serviço SDK: RdsDataClient	Cliente de serviço SDK: RdsClient
Exemplo: aplicativo React/Spring REST usando RdsDataClient	Exemplos: Vários RdsClient exemplos

Exemplos Amazon DocumentDB

Trabalhar com dados	Trabalhar com a base de dados
API não SDK: biblioteca Java específica do MongoDB (por exemplo, MongoDB para Java); seu código gerencia conexões de banco de dados ou um pool de conexões.	Cliente de serviço SDK: DocDbClient
Exemplos: Guia do desenvolvedor do DocumentDB (Mongo) (selecione a guia 'Java')	

Trabalhar com DynamoDB

Esta seção fornece exemplos que mostram como trabalhar com o [DynamoDB](#). Os exemplos nesta seção usam o cliente DynamoDB padrão de baixo nível ([DynamoDbClient](#)) oferecido com o 2.x AWS SDK for Java

O SDK também oferece o [DynamoDB Enhanced Client](#), que fornece uma abordagem de alto nível e orientada a objetos para trabalhar com o DynamoDB.

Tópicos

- [Trabalhe com tabelas em DynamoDB](#)
- [Trabalhe com itens em DynamoDB](#)

Trabalhe com tabelas em DynamoDB

Tabelas são os contêineres de todos os itens em um banco de dados do DynamoDB. Para adicionar ou remover dados do DynamoDB, você deve criar uma tabela.

Para cada tabela, você deve definir:

- Um nome de tabela é exclusivo para a conta e a região.
- Uma chave primária para a qual cada valor deve ser único; dois itens na tabela não podem ter o mesmo valor de chave primária.

Uma chave primária pode ser simples, consistindo em uma única chave de partição (HASH) ou composta, que consiste em uma partição e uma chave de classificação (RANGE).

Cada valor chave tem um tipo de dados associado, enumerado pela classe. [ScalarAttributeType](#) O valor da chave pode ser binário (B), numérico (N) ou uma string (S). Para obter mais informações, consulte [Regras de nomenclatura e tipos de dados](#) no Guia do Amazon DynamoDB desenvolvedor.

- Throughput provisionado são valores que definem o número de unidades de capacidade de leitura/gravação reservado para a tabela.

 Note

Amazon DynamoDB o [preço](#) é baseado nos valores de taxa de transferência provisionados que você define em suas tabelas, portanto, reserve apenas a capacidade que você acha que precisará para sua tabela.

O throughput provisionado para uma tabela pode ser modificado a qualquer momento. Dessa forma, você poderá ajustar a capacidade conforme suas necessidades mudam.

Criar uma tabela

Use o método `createTable` do `DynamoDbClient` para criar uma tabela do DynamoDB. Você precisa construir atributos de tabela e um esquema de tabela, ambos usados para identificar a chave primária da tabela. Você também deve fornecer valores de throughput provisionado iniciais e um nome de tabela.

Note

Se uma tabela com o nome que você escolheu já existir, [DynamoDbException](#) será lançada.

Criar uma tabela com uma chave primária simples

Esse código cria uma tabela com um atributo que é a chave primária simples da tabela. O exemplo usa [AttributeDefinition](#) e [KeySchemaElement](#) objetos para o [CreateTableRequest](#)

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) em GitHub.

Criar uma tabela com uma chave primária composta

O exemplo a seguir cria uma tabela com dois atributos. Ambos os atributos são usados para a chave primária composta.

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) em GitHub.

Listar tabelas

Você pode listar as tabelas em uma determinada região chamando o método `listTables` do `DynamoDbClient`.



Note

Se a tabela nomeada não existir para sua conta e região, um [`ResourceNotFoundException`](#) será lançado.

Importações

```
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;
```

Código

```
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!");
}
```

Por padrão, até 100 tabelas são retornadas por chamada — use `lastEvaluatedTableName` no [ListTablesResponse](#) objeto retornado para obter a última tabela que foi avaliada. Você pode usar esse valor para iniciar a listagem depois do último valor retornado da listagem anterior.

Veja o [exemplo completo](#) em GitHub.

Descrever (obter informações sobre) uma tabela

Use o `DynamoDbClient's describeTable` método para obter informações sobre uma tabela.

Note

Se a tabela nomeada não existir para sua conta e região, um [ResourceNotFoundException](#) será lançado.

Importações

```
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;
```

Código

```
public static void describeDynamoDBTable(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!");
}
```

Veja o [exemplo completo](#) emGitHub.

Modificar (atualizar) uma tabela

Você pode modificar os valores de taxa de transferência provisionados da sua tabela a qualquer momento chamando o método `DynamoDbClient's updateTable`

Note

Se a tabela nomeada não existir para sua conta e região, um [`ResourceNotFoundException`](#) será lançado.

Importações

```
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;
```

Código

```
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!");
}
```

Veja o [exemplo completo](#) em GitHub.

Excluir uma tabela

Para excluir uma tabela, chame o `DynamoDbClient's deleteTable` método e forneça o nome da tabela.

Note

Se a tabela nomeada não existir para sua conta e região, um [ResourceNotFoundException](#) será lançado.

Importações

```
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;
```

Código

```
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!");
}
```

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [Diretrizes para trabalhar com tabelas](#) no Guia do Amazon DynamoDB desenvolvedor
- [Trabalhando com tabelas DynamoDB](#) no Guia do Amazon DynamoDB Desenvolvedor

Trabalhe com itens em DynamoDB

No DynamoDB, um item é um conjunto de atributos, e cada um tem um nome e um valor. Um valor de atributo pode ser uma escalar, um conjunto ou um tipo de documento. Para obter mais informações, consulte [Regras de nomenclatura e tipos de dados](#) no Guia do Amazon DynamoDB desenvolvedor.

Recuperar (obter) um item de uma tabela

Chame o `getItem` método `DynamoDbClient`'s e passe a ele um [GetItemRequest](#) objeto com o nome da tabela e o valor da chave primária do item que você deseja. Ele retorna um [GetItemResponse](#) objeto com todos os atributos desse item. Você pode especificar uma ou mais [expressões de projeção](#) no `GetItemRequest` para recuperar atributos específicos.

Você pode usar o `item()` método do `GetItemResponse` objeto retornado para recuperar um [mapa](#) dos pares de chave (`String AttributeValue`) e valor () associados ao item.

Importações

```
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;
```

Código

```
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);
}
}
```

Veja o [exemplo completo](#) emGitHub.

Recuperar (Get) um item de uma tabela usando o cliente assíncrono

Invoque o `getItem` método do `DynamoDbAsyncClient` e passe a ele um [`GetItemRequest`](#) objeto com o nome da tabela e o valor da chave primária do item que você deseja.

Você pode retornar uma instância de [`Collection`](#) com todos os atributos desse item (consulte o exemplo a seguir).

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Adicionar um novo item a uma tabela

Crie um [Mapa](#) de pares de chave/valor que representem os atributos do item. Eles devem incluir valores para os campos de chave primária da tabela. Se o item identificado pela chave primária já existir, os campos serão atualizados pela requisição.

Note

Se a tabela nomeada não existir para sua conta e região, um [ResourceNotFoundException](#) será lançado.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Atualizar um item existente em uma tabela

Você pode atualizar um atributo para um item já existente em uma tabela usando o método `updateItem` do `DynamoDbClient`, fornecendo um nome de tabela, o valor da chave primária e um mapa de campos a ser atualizado.



Note

Se a tabela nomeada não existir para sua conta e região, ou se o item identificado pela chave primária que você inseriu não existir, `ResourceNotFoundException` será lançado.

Importações

```
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;
```

Código

```
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!");
}
```

Veja o [exemplo completo](#) em GitHub.

Excluir um item existente em uma tabela

Você pode excluir um item que existe em uma tabela usando o método `deleteItem` do `DynamoDbClient` e fornecendo um nome de tabela, bem como o valor da chave primária.

Note

Se a tabela nomeada não existir para sua conta e região, ou se o item identificado pela chave primária que você inseriu não existir, [ResourceNotFoundException](#) será lançado.

Importações

```
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;
```

Código

```
public static void deleteDynamoDBItem(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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [Diretrizes para trabalhar com itens](#) no Guia do Amazon DynamoDB desenvolvedor
- [Trabalhando com itens DynamoDB](#) no Guia do Amazon DynamoDB Desenvolvedor

Trabalhar com Amazon EC2

Esta seção fornece exemplos de programação [Amazon EC2](#)que usam o AWS SDK for Java 2.x.

Tópicos

- [Gerenciar Amazon EC2 instâncias](#)
- [Use endereços IP elásticos em Amazon EC2](#)
- [Zonas Regiões da AWS de uso e disponibilidade](#)
- [Trabalhar comAmazon EC2Pares de chaves do](#)
- [Trabalhar com grupos de segurança noAmazon EC2](#)
- [Trabalhe com metadados da instância do Amazon EC2](#)

Gerenciar Amazon EC2 instâncias

Criar uma instância do

Crie uma nova Amazon EC2 instância chamando o [runInstances](#)método do [Ec2Client](#), fornecendo a ela uma [imagem de RunInstancesRequestmáquina da Amazon \(AMI\)](#) para usar e um tipo de [instância](#).

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) emGitHub.

Iniciar uma instância

Para iniciar uma Amazon EC2 instância, chame o [startInstances](#)método do Ec2Client, fornecendo a ele um [StartInstancesRequest](#)contendo o ID da instância a ser iniciada.

Importações

```
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;
```

Código

```
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);
}
```

Veja o [exemplo completo](#) emGitHub.

Interromper uma instância

Para interromper uma Amazon EC2 instância, chame o [stopInstances](#)método do Ec2Client, fornecendo a ele um [StopInstancesRequest](#)contendo o ID da instância a ser interrompida.

Importações

```
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;
```

Código

```
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);  
}
```

Veja o [exemplo completo](#) emGitHub.

Reiniciar uma instância

Para reiniciar uma Amazon EC2 instância, chame o [rebootInstances](#) método do Ec2Client, fornecendo a ele um [RebootInstancesRequest](#) contendo o ID da instância a ser reinicializada.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Descrever instâncias

Para listar suas instâncias, crie um [DescribeInstancesRequest](#) chame o método do Ec2Client. [describeInstances](#) Ele retornará um [DescribeInstancesResponse](#) objeto que você pode usar para listar as Amazon EC2 instâncias da sua conta e região.

As instâncias são agrupadas por reserva. Cada reserva corresponde à chamada a `startInstances` que iniciou a instância. Para listar as instâncias, você deve primeiro chamar o método `reservations` da classe `DescribeInstancesResponse` e chamar `instances` em cada objeto [Reservation](#) retornado.

Importações

```
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;
```

Código

```
public static void describeEC2Instances( Ec2Client ec2){

    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);
}
}
```

Os resultados são paginados; você pode obter mais resultados passando o valor retornado do método `nextToken` do objeto de resultado para o método `nextToken` do objeto de uma solicitação nova e usando o objeto da nova solicitação na próxima chamada para `describeInstances`.

Veja o [exemplo completo](#) emGitHub.

Monitorar uma instância

Você pode monitorar diversos aspectos das instâncias do Amazon EC2, como utilização de CPU e rede, memória disponível e espaço em disco restante. Para saber mais sobre monitoramento de instância, consulte [Monitoramento Amazon EC2](#) no Guia do Amazon EC2 usuário para instâncias do Linux.

Para começar a monitorar uma instância, você deve criar um [`MonitorInstancesRequest`](#) com o ID da instância a ser monitorado e passá-lo para o método do `Ec2Client`. [`monitorInstances`](#)

Importações

```
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;
```

Código

```
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);  
}
```

Veja o [exemplo completo](#) emGitHub.

Interromper o monitoramento de instâncias

Para interromper o monitoramento de uma instância, crie um [UnmonitorInstancesRequest](#) com o ID da instância para interromper o monitoramento e passe-o para o método do Ec2Client. [unmonitorInstances](#)

Importações

```
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;
```

Código

```
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);
```

}

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [RunInstances](#)na referência da Amazon EC2 API
- [DescribeInstances](#)na referência da Amazon EC2 API
- [StartInstances](#)na referência da Amazon EC2 API
- [StopInstances](#)na referência da Amazon EC2 API
- [RebootInstances](#)na referência da Amazon EC2 API
- [MonitorInstances](#)na referência da Amazon EC2 API
- [UnmonitorInstances](#)na referência da Amazon EC2 API

Use endereços IP elásticos em Amazon EC2

Alocar um endereço IP elástico

Para usar um endereço IP elástico, você primeiro aloca um para sua conta e o associa à instância ou a uma interface de rede.

Para alocar um endereço IP elástico, chame o `allocateAddress` método do `Ec2Client` com um [AllocateAddressRequest](#) objeto contendo o tipo de rede (clássico Amazon EC2 ou VPC).

O retornado [AllocateAddressResponse](#) contém um ID de alocação que você pode usar para associar o endereço a uma instância, passando o ID de alocação e o ID da instância em [AssociateAddressRequest](#) para o método do `Ec2Client.associateAddress`

Importações

```
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;
```

Código

```
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 "";  
}
```

Veja o [exemplo completo](#) em GitHub.

Descrever endereços IP elásticos

Para listar os endereços IP elásticos atribuídos à sua conta, chame o método do Ec2Client `describeAddresses`. Ele retorna um [DescribeAddressesResponse](#) que você pode usar para obter uma lista de objetos de [endereço](#) que representam os endereços IP elásticos em sua conta.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Liberar um endereço IP elástico

Para liberar um endereço IP elástico, chame o `releaseAddress` método do `Ec2Client`, passando a ele um [`ReleaseAddressRequest`](#) contendo o ID de alocação do endereço IP elástico que você deseja liberar.

Importações

```
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;
```

Código

```
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);  
    }  
}
```

Depois de liberar um endereço IP elástico, ele é liberado para o pool de endereços AWS IP e pode ficar indisponível para você posteriormente. Não se esqueça de atualizar os registros DNS e todos os servidores ou dispositivos que se comunicam com o endereço.

Se você estiver usando o EC2-Classic ou uma VPC padrão, liberar um endereço IP elástico o desassociará automaticamente de qualquer instância a qual esteja associada. Para desassociar um endereço IP elástico sem liberá-lo, use o método do Ec2Client. `disassociateAddress`

Se estiver usando uma VPC não padrão, você deverá usar `disassociateAddress` para desassociar o endereço IP elástico antes de tentar liberá-lo. Caso contrário, Amazon EC2 retornará um erro (`InvalidIPAddress. InUse`).

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [Endereços IP elásticos](#) no Guia Amazon EC2 do usuário para instâncias Linux
- [AllocateAddress](#)na referência da Amazon EC2 API
- [DescribeAddresses](#)na referência da Amazon EC2 API
- [ReleaseAddress](#)na referência da Amazon EC2 API

Zonas Regiões da AWS de uso e disponibilidade

Descrever as regiões

Para listar as regiões disponíveis para sua conta, chame o método do Ec2Client.

`describeRegions` Ele retorna um [DescribeRegionsResponse](#). Chame o método `regions` do objeto retornado para obter uma lista de objetos [Region](#) que representam cada região.

Importações

```
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;
```

Código

```
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();
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Descrever zonas de disponibilidade

Para listar cada zona de disponibilidade disponível para sua conta, chame o método do Ec2Client.

`describeAvailabilityZones` Ele retorna um [DescribeAvailabilityZonesResponse](#). Chame seu `availabilityZones` método para obter uma lista de [AvailabilityZone](#) objetos que representam cada zona de disponibilidade.

Importações

```
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;
```

Código

Crie o 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();
```

Em seguida, chame describeAvailabilityZones () e recupere os resultados.

```
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();
```

Veja o [exemplo completo](#) emGitHub.

Descrever contas

Para listar informações relacionadas ao EC2 sobre sua conta, chame o método do Ec2Client. `describeAccountAttributes` Esse método retorna um [`DescribeAccountAttributesResponse`](#) objeto. Invoque esse `accountAttributes` método de objetos

para obter uma lista de [AccountAttribute](#) objetos. Você pode percorrer a lista para recuperar um objeto [AccountAttribute](#).

Você pode obter os valores dos atributos da sua conta invocando o `attributeValues` método do [AccountAttribute](#) objeto. Esse método retorna uma lista de [AccountAttributeValue](#) objetos. É possível percorrer essa segunda lista para exibir o valor dos atributos (veja o exemplo de código a seguir).

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [Regiões e zonas de disponibilidade](#) no Guia Amazon EC2 do usuário para instâncias Linux
- [DescribeRegions](#)na referência da Amazon EC2 API
- [DescribeAvailabilityZones](#)na referência da Amazon EC2 API

Trabalhar comAmazon EC2Pares de chaves do

Criar um par de chaves

Para criar um key pair, chame o `Ec2Client.createKeyPair`Método do com um[CreateKeyPairRequest](#)que contém o nome da chave.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) no GitHub.

Descrever pares de chaves

Para listar os pares de chaves ou obter informações sobre eles, chame o `Ec2Client.describeKeyPairs` Método do. Ele retorna um [DescribeKeyPairsResponse](#) que pode ser usado para acessar a lista de pares de chaves chamando o método `keyPairs`, que retorna uma lista de objetos [KeyPairInfo](#).

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) no GitHub.

Excluir um par de chaves

Para excluir um key pair, chame o `Ec2ClientdeleteKeyPair`método, passando um[DeleteKeyPairRequest](#)que contenha o nome do key pair a ser excluído.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) no GitHub.

Mais informações

- [Amazon EC2Pares de chaves](#) no Amazon EC2User Guide for Linux Instances
- [CreateKeyPair](#) no Amazon EC2Referência de API do
- [DescribeKeyPairs](#) no Amazon EC2Referência de API do
- [DeleteKeyPair](#) no Amazon EC2Referência de API do

Trabalhar com grupos de segurança noAmazon EC2

Crie um grupo de segurança

Para criar um grupo de segurança, chame o `Ec2Client.createSecurityGroup` Método do com um [CreateSecurityGroupRequest](#) que contém o nome da chave.

Importações

```
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;
```

Código

```
    CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
    .groupName(groupName)
    .description(groupDesc)
    .vpcId(vpcId)
    .build();

    CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

Veja o [exemplo completo](#) no GitHub.

Configurar um grupo de segurança

Um grupo de segurança pode controlar os tráfegos de entrada e saída para as instâncias do Amazon EC2.

Para adicionar regras de entrada ao grupo de segurança, use o `Ec2Client.authorizeSecurityGroupIngress` Método do, fornecendo o nome do grupo de segurança e as regras de acesso ([IpPermission](#)) que você deseja atribuir a ele dentro de

um [AuthorizeSecurityGroupIngressRequest](#) objeto. O exemplo a seguir mostra como adicionar permissões de IP a um grupo de segurança.

Importações

```
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;
```

Código

Primeiro, crie um Ec2Client

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

Em seguida, use o Ec2Client authorizeSecurityGroupIngress Método do,

```
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 "";
}
```

Para adicionar uma regra de saída ao grupo de segurança, forneça dados semelhantes em um [AuthorizeSecurityGroupEgressRequest](#) para o `Ec2ClientauthorizeSecurityGroupEgress`Método do.

Veja o [exemplo completo](#) no GitHub.

Descrever grupos de segurança

Para descrever os grupos de segurança ou obter informações sobre eles, chame o `Ec2ClientdescribeSecurityGroups`Método do. Ele retorna um [DescribeSecurityGroupsResponse](#) que pode ser usado para acessar a lista de grupos de segurança chamando o método `securityGroups`, que retorna uma lista de objetos [SecurityGroup](#).

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) no GitHub.

Excluir um grupo de segurança

Para excluir um grupo de segurança, chame o `Ec2Client.deleteSecurityGroup` método, passando um [DeleteSecurityGroupRequest](#) que contenha o ID do security group a ser excluído.

Importações

```
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;
```

Código

```
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);  
    }  
}
```

Veja o [exemplo completo](#) no GitHub.

Mais informações

- [Amazon EC2Security groups](#) no [Amazon EC2User Guide for Linux Instances](#)
- [Autorizar tráfego de entrada para as instâncias do Linux](#) no [Amazon EC2User Guide for Linux Instances](#)
- [CreateSecurityGroup](#) no [Amazon EC2Referência de API do](#)
- [DescribeSecurityGroups](#) no [Amazon EC2Referência de API do](#)
- [DeleteSecurityGroup](#) no [Amazon EC2Referência de API do](#)
- [AuthorizeSecurityGroupIngress](#) no [Amazon EC2Referência de API do](#)

Trabalhe com metadados da instância do Amazon EC2

Um cliente Java SDK para o Amazon EC2 Instance Metadata Service (cliente de metadados) permite que seus aplicativos acessem metadados em sua instância EC2 local. O cliente de metadados trabalha com a instância local do [IMDSv2](#) (Instance Metadata Service v2) e usa solicitações orientadas por sessão.

Duas classes de cliente estão disponíveis no SDK. O síncrono [Ec2MetadataClient](#) é para operações de bloqueio e o [Ec2MetadataAsyncClient](#) é para casos de uso assíncronos e sem bloqueio.

Conceitos básicos

Para usar o cliente de metadados, adicione o artefato `imds` Maven ao seu projeto. Você também precisa de classes para an [SdkHttpClient](#) (ou an [SdkAsyncHttpClient](#) para a variante assíncrona) no classpath.

O XML Maven a seguir mostra trechos de dependência para usar o síncrono [URLConnectionHttpClient](#) junto com a dependência para clientes de metadados.

```
<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>
```

Pesquise no [repositório central do Maven](#) a versão mais recente do bom artefato.

Para usar um cliente HTTP assíncrono, substitua o trecho de dependência do `url-connection-client` artefato. Por exemplo, o trecho a seguir traz a [NettyNioAsyncHttpClient](#) implementação.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>netty-nio-client</artifactId>
</dependency>
```

Use o cliente de metadados

Instanciar um cliente de metadados

Você pode instanciar uma instância de um síncrono `Ec2MetadataClient` quando somente uma implementação da `SdkHttpClient` interface está presente no classpath. Para fazer isso, chame o `Ec2MetadataClient#create()` método estático, conforme mostrado no trecho a seguir.

```
Ec2MetadataClient client = Ec2MetadataClient.create(); //
'Ec2MetadataAsyncClient#create' is the asynchronous version.
```

Se seu aplicativo tiver várias implementações da `SdkHttpAsyncClient` interface `SdkHttpClient` ou, você deverá especificar uma implementação para o cliente de metadados usar, conforme mostrado na [seção chamada “Cliente HTTP configurável”](#).

Note

Para a maioria dos clientes de serviços, como o Amazon S3, o SDK for Java adiciona automaticamente implementações da `SdkHttpAsyncClient` interface `SdkHttpClient` ou. Se o seu cliente de metadados usar a mesma implementação, `Ec2MetadataClient#create()` ele funcionará. Se você precisar de uma implementação diferente, deverá especificá-la ao criar o cliente de metadados.

Enviar solicitações

Para recuperar metadados da instância, instancie a `EC2MetadataClient` classe e chame o `get` método com um parâmetro de caminho que especifica a [categoria de metadados da instância](#).

O exemplo a seguir imprime o valor associado à `ami-id` chave no console.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/ami-id");
System.out.println(responseasString());
```

```
client.close(); // Closes the internal resources used by the Ec2MetadataClient class.
```

Se o caminho não for válido, o get método gerará uma exceção.

Reutilize a mesma instância do cliente para várias solicitações, mas chame close o cliente quando não for mais necessário liberar recursos. Depois que o método close é chamado, a instância do cliente não pode mais ser usada.

Analise as respostas

Os metadados da instância do EC2 podem ser produzidos em diferentes formatos. Texto sem formatação e JSON são os formatos mais usados. Os clientes de metadados oferecem maneiras de trabalhar com esses formatos.

Como mostra o exemplo a seguir, use o asString método para obter os dados como uma string Java. Você também pode usar o asList método para separar uma resposta em texto sem formatação que retorne várias linhas.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

Se a resposta estiver em JSON, use o Ec2MetadataResponse#asDocument método para analisar a resposta JSON em uma instância de [documento](#), conforme mostrado no trecho de código a seguir.

```
Document fullResponse = response.asDocument();
```

Uma exceção será lançada se o formato dos metadados não estiver em JSON. Se a resposta for analisada com êxito, você poderá usar a [API do documento](#) para inspecionar a resposta com mais detalhes. Consulte o [gráfico de categorias de metadados](#) da instância para saber quais categorias de metadados fornecem respostas no formato JSON.

Configurar um cliente de metadados

Tentativas

Você pode configurar um cliente de metadados com um mecanismo de repetição. Se você fizer isso, o cliente poderá repetir automaticamente as solicitações que falharem por motivos inesperados. Por

padrão, o cliente tenta novamente três vezes em uma solicitação com falha, com um tempo de atraso exponencial entre as tentativas.

Se seu caso de uso exigir um mecanismo de repetição diferente, você poderá personalizar o cliente usando o `retryPolicy` método em seu construtor. Por exemplo, o exemplo a seguir mostra um cliente síncrono configurado com um atraso fixo de dois segundos entre as tentativas e cinco tentativas de repetição.

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
    retryPolicyBuilder.numRetries(5)

    .backoffStrategy(fixedBackoffStrategy))
        .build();
```

Há vários [BackoffStrategies](#) que você pode usar com um cliente de metadados.

Você também pode desativar totalmente o mecanismo de repetição, como mostra o trecho a seguir.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(Ec2MetadataRetryPolicy.none())
        .build();
```

O uso `Ec2MetadataRetryPolicy#none()` desativa a política de repetição padrão para que o cliente de metadados não tente novamente.

Versão IP

Por padrão, um cliente de metadados usa o endpoint IPV4 em `http://169.254.169.254`. Para alterar o cliente para usar a versão IPV6, use o método `endpointMode` ou o `endpoint` método do construtor. Uma exceção ocorre se os dois métodos forem chamados no construtor.

Os exemplos a seguir mostram as duas opções de IPV6.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpointMode(EndpointMode.IPV6)
```

```
.build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .endpoint(URI.create("http://[fd00:ec2::254]"))
        .build();
```

Recursos principais

Cliente assíncrono

Para usar a versão sem bloqueio do cliente, instancie uma instância da `Ec2MetadataAsyncClient` classe. O código no exemplo a seguir cria um cliente assíncrono com configurações padrão e usa o `get` método para recuperar o valor da `ami-id` chave.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/
ami-id");
```

O `java.util.concurrent.CompletableFuture` retornado pelo `get` método é concluído quando a resposta retorna. O exemplo a seguir imprime os `ami-id` metadados no console.

```
response.thenAccept(metadata -> System.out.println(metadataasString()));
```

Cliente HTTP configurável

O construtor de cada cliente de metadados tem um `httpClient` método que você pode usar para fornecer um cliente HTTP personalizado.

O exemplo a seguir mostra o código de uma `URLConnectionHttpClient` instância personalizada 1.

```
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.
```

O exemplo a seguir mostra o código de uma Netty Nio Async Http Client instância personalizada com um cliente de metadados assíncrono.

```
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.
```

O [the section called “Clientes HTTP”](#) tópico deste guia fornece detalhes sobre como configurar os clientes HTTP que estão disponíveis no SDK for Java.

Cache de tokens

Como os clientes de metadados usam o IMDSv2, todas as solicitações são associadas a uma sessão. Uma sessão é definida por um token que tem uma expiração, que o cliente de metadados gerencia para você. Cada solicitação de metadados reutiliza automaticamente o token até que ele expire.

Por padrão, um token dura seis horas (21.600 segundos). Recomendamos que você mantenha o time-to-live valor padrão 1. Recomendamos que você mantenha o valor padrão 1.

Se necessário, configure a duração usando o método `tokenTtl` builder. Por exemplo, o código no trecho a seguir cria um cliente com uma duração de sessão de cinco minutos.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .tokenTtl(Duration.ofMinutes(5))
        .build();
```

Se você omitir a chamada `dotokenTtl` método no construtor, a duração padrão de 21.600 será usada em vez disso.

Trabalhar com IAM

Esta seção fornece exemplos de programação [IAM](#) usando o AWS SDK for Java 2.x.

AWS Identity and Access Management(IAM) permite que você controle com segurança o acesso aos AWS serviços e recursos para seus usuários. Usando IAM, você pode criar e gerenciar AWS usuários e grupos e usar permissões para permitir e negar o acesso deles aos AWS recursos. Para obter um guia completo sobre IAM, visite o [Guia IAM do usuário](#).

Os exemplos a seguir incluem somente o código necessário para demonstrar cada técnica. O [código de exemplo completo está disponível em GitHub](#). A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

- [Gerenciar chaves de IAM acesso](#)
- [Gerenciar IAM usuários](#)
- [Use aliases de IAM conta](#)
- [Trabalhe com IAM políticas](#)
- [Trabalhe com certificados IAM de servidor](#)

Gerenciar chaves de IAM acesso

Criar uma chave de acesso

Para criar uma chave de IAM acesso, chame o `createAccessKey` método `IamClient's` com um [CreateAccessKeyRequest](#) objeto.

 Note

Você deve definir a região como `AWS_GLOBAL` para chamadas do `IamClient` para trabalhar, porque o IAM é um serviço global.

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) em GitHub.

Listar chaves de acesso

Para listar as chaves de acesso de um determinado usuário, crie um [ListAccessKeysRequest](#) objeto que contenha o nome de usuário para o qual listar as chaves e passe-o para o `listAccessKeys` método `IamClient's`.

Note

Se você não fornecer um nome de usuário para `listAccessKeys`, ele tentará listar as chaves de acesso associadas à pessoa Conta da AWS que assinou a solicitação.

Importações

```
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;
```

Código

```
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);
    }
}
```

Os resultados de `listAccessKeys` são paginados (com um máximo de 100 registros por chamada). Você pode chamar `isTruncated` o [ListAccessKeysResponse](#) objeto retornado para ver se a consulta retornou menos resultados do que os disponíveis. Se esse for o caso, chame `marker` no `ListAccessKeysResponse` e use-o ao criar uma nova solicitação. Use essa nova solicitação na próxima invocação de `listAccessKeys`.

Veja o [exemplo completo](#) em GitHub.

Recuperar a hora do uso mais recente de uma chave de acesso

Para saber a hora em que uma chave de acesso foi usada pela última vez, chame o `getAccessKeyLastUsed` método `IamClient's` com o ID da chave de acesso (que pode ser passado usando um [GetAccessKeyLastUsedRequest](#) objeto).

Em seguida, você pode usar o [GetAccessKeyLastUsedResponse](#) objeto retornado para recuperar o último tempo de uso da chave.

Importações

```
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;
```

Código

```
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");  
}
```

Veja o [exemplo completo](#) em GitHub.

Ativar ou desativar chaves de acesso

Você pode ativar ou desativar uma chave de acesso criando um [UpdateAccessKeyRequest](#) objeto, fornecendo o ID da chave de acesso, opcionalmente o nome do usuário e o [status](#) desejado e, em seguida, passando o objeto de solicitação para o método `IamClient'supdateAccessKey`.

Importações

```
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;
```

Código

```
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);
}
}
```

Veja o [exemplo completo](#) em GitHub.

Excluir uma chave de acesso

Para excluir permanentemente uma chave de acesso, chame o `deleteKey` método `IamClient`'s, fornecendo a ele um [DeleteAccessKeyRequest](#) contendo o ID e o nome de usuário da chave de acesso.

Note

Depois de excluída, uma chave não poderá mais ser recuperada ou usada. Para desativar temporariamente uma chave para que ela possa ser ativada novamente mais tarde, use o [updateAccessKey](#) método em vez disso.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [CreateAccessKey](#)na referência da IAM API
- [ListAccessKeys](#)na referência da IAM API
- [GetAccessKeyLastUsed](#)na referência da IAM API
- [UpdateAccessKey](#)na referência da IAM API
- [DeleteAccessKey](#)na referência da IAM API

Gerenciar IAM usuários

Criar um usuário

Crie um novo IAM usuário fornecendo o nome de usuário ao `createUser` método `IamClient`'s usando um [CreateUserRequest](#)objeto contendo o nome do usuário.

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) emGitHub.

Listar usuários do

Para listar os IAM usuários da sua conta, crie um novo [ListUsersRequest](#) e passe-o para o `listUsers` método `IamClient`'s. Você pode recuperar a lista de usuários `users` chamando o [ListUsersResponse](#) objeto retornado.

A lista de usuários retornados por `listUsers` é paginada. Você pode verificar se há mais resultados a serem recuperados chamando o método `isTruncated` do objeto de resposta. Se ele retornar `true`, chame o método `marker()` do objeto de resposta. Use o valor do marcador para criar um novo objeto de solicitação. Em seguida, chame o método `listUsers` novamente com a nova solicitação.

Importações

```
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;
```

Código

```
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);
}
}
```

Veja o [exemplo completo](#) em GitHub.

Atualizar um usuário

Para atualizar um usuário, chame o `updateUser` método do `IamClient` objeto, que usa um [`UpdateUserRequest`](#) objeto que você pode usar para alterar o nome ou o caminho do usuário.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Excluir um usuário

Para excluir um usuário, chame a `deleteUser` solicitação `IamClient's` com um [`UpdateUserRequest`](#) objeto definido com o nome de usuário a ser excluído.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [IAM Usuários](#) no Guia IAM do Usuário
- [Gerenciando IAM usuários](#) no Guia IAM do usuário
- [CreateUser](#) na referência da IAM API
- [ListUsers](#) na referência da IAM API

- [UpdateUser](#)na referência da IAM API
- [DeleteUser](#)na referência da IAM API

Use aliases de IAM conta

Se você quiser que o URL da sua página de login contenha o nome da sua empresa ou outro identificador amigável em vez do seu Conta da AWS ID, você pode criar um alias para o seu Conta da AWS



Note

AWS suporta exatamente um alias de conta por conta.

Criar um alias de conta

Para criar um alias de conta, chame o `createAccountAlias` método `IamClient's` com um [`CreateAccountAliasRequest`](#) objeto que contém o nome do alias.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Listar aliases de conta

Para listar o alias da conta, se houver, chame o método `listAccountAliases` do `IamClient`.

Note

O retornado [`ListAccountAliasesResponse`](#) suporta os mesmos marker métodos `isTruncated` e métodos de outros métodos AWS SDK for Java de lista, mas uma conta só pode ter um alias de conta.

Importações

```
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;
```

Código

```
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);
    }
}
```

```
}
```

veja o [exemplo completo](#) emGitHub.

Excluir um alias de conta

Para excluir o alias da conta, chame o método `deleteAccountAlias` do `IamClient`. Ao excluir um alias de conta, você deve fornecer seu nome usando um [`DeleteAccountAliasRequest`](#) objeto.

Importações

```
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;
```

Código

```
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");
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [Seu ID de AWS conta e seu alias](#) no Guia do IAM usuário
- [CreateAccountAlias](#)na referência da IAM API

- [ListAccountAliases](#)na referência da IAM API
- [DeleteAccountAlias](#)na referência da IAM API

Trabalhe com IAM políticas

Criar uma política

Para criar uma nova política, forneça o nome da política e um documento de política formatado em JSON no método [CreatePolicyRequest](#)to the. IamClient createPolicy

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) em GitHub.

Obter uma política

Para recuperar uma política existente, chame o `getPolicy` método `IamClient`'s, fornecendo o ARN da política em um [`GetPolicyRequest`](#) objeto.

Importações

```
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;
```

Código

```
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);
    }
}
```

```
    }  
}
```

Veja o [exemplo completo](#) em GitHub.

Anexar uma política de função

Você pode anexar uma política a uma IAM [função](#) chamando o `attachRolePolicy` método `IamClient's`, fornecendo a ele o nome da função e o ARN da política em um [AttachRolePolicyRequest](#).

Importações

```
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;
```

Código

```
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");
}
```

Veja o [exemplo completo](#) emGitHub.

Listar políticas de função anexadas

Liste as políticas anexadas em uma função chamando o método `listAttachedRolePolicies` do `IamClient`. É preciso um [`ListAttachedRolePoliciesRequest`](#) objeto que contém o nome da função para listar as políticas.

Chame `getAttachedPolicies` o [`ListAttachedRolePoliciesResponse`](#) objeto retornado para obter a lista de políticas anexadas. Os resultados podem estar truncados; se o método `ListAttachedRolePoliciesResponse` do objeto `isTruncated` retornar `true`, chame o método `ListAttachedRolePoliciesResponse` do objeto `marker`. Use o marcador retornado para criar uma nova solicitação e use-o para chamar `listAttachedRolePolicies` novamente a fim de obter o próximo lote de resultados.

Importações

```
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;
```

Código

```
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");  
}
```

Veja o [\[exemplo completo\]](https://github.com/awsdocs/aws-doc-sdk-examples/blob/master/javav2/example-code-iam/src/main/java/com/example/iam/AttachRolePolicy.java) em GitHub

Desanexar uma política de função

Para separar uma política de uma função, chame o `detachRolePolicy` método `IamClient`'s, fornecendo a ele o nome da função e o ARN da política em a. [DetachRolePolicyRequest](#)

Importações

```
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;
```

Código

```
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);
        }
    }
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [Visão geral das IAM políticas](#) no Guia IAM do usuário.
- [AWS IAM Referência de política](#) no Guia IAM do usuário.
- [CreatePolicy](#)na referência da IAM API
- [GetPolicy](#)na referência da IAM API
- [AttachRolePolicy](#)na referência da IAM API
- [ListAttachedRolePolicies](#)na referência da IAM API
- [DetachRolePolicy](#)na referência da IAM API

Trabalhe com certificados IAM de servidor

Para habilitar conexões HTTPS com seu site ou aplicativo emAWS, você precisa de um certificado de servidor SSL/TLS. Você pode usar um certificado de servidor fornecido pelo AWS Certificate Manager ou um obtido junto a um provedor externo.

Recomendamos usar o ACM para provisionar, gerenciar e implantar os certificados de servidor. Com ACM você, você pode solicitar um certificado, implantá-lo em seus AWS recursos e deixar que você ACM gerencie as renovações de certificados. Os certificados fornecidos pelo ACM são gratuitos.

Para obter mais informações sobre o ACM, consulte o [Guia do usuário do AWS Certificate Manager](#).

Obter um certificado do servidor

Você pode recuperar um certificado de servidor chamando o `getServerCertificate` método `IamClient's`, passando-o a [GetServerCertificateRequest](#)com o nome do certificado.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Listar certificados do servidor

Para listar seus certificados de servidor, chame o `listServerCertificates` método `IamClient's` com um [`ListServerCertificatesRequest`](#). Ele retorna um [`ListServerCertificatesResponse`](#).

Chame o `serverCertificateMetadataList` método do `ListServerCertificateResponse` objeto retornado para obter uma lista de [`ServerCertificateMetadata`](#) objetos que você pode usar para obter informações sobre cada certificado.

Os resultados podem estar truncados; se o método `ListServerCertificateResponse` do objeto `isTruncated` retornar `true`, chame o método `ListServerCertificatesResponse` do objeto `marker` e use o marcador para criar uma solicitação. Use a nova solicitação para chamar `listServerCertificates` novamente a fim de obter o próximo lote de resultados.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Atualizar um certificado do servidor

Você pode atualizar o nome ou o caminho de um certificado de servidor chamando o método `updateServerCertificate` do `IamClient`. Ele usa um conjunto de [`UpdateServerCertificateRequest`](#) objetos com o nome atual do certificado do servidor e um novo nome ou um novo caminho para usar.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Excluir um certificado do servidor

Para excluir um certificado de servidor, chame o `deleteServerCertificate` método `IamClient's` com a [DeleteServerCertificateRequest](#)contendo o nome do certificado.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Mais informações

- [Trabalhando com certificados de servidor](#) no Guia IAM do usuário

- [GetServerCertificate](#)na referência da IAM API
- [ListServerCertificates](#)na referência da IAM API
- [UpdateServerCertificate](#)na referência da IAM API
- [DeleteServerCertificate](#)na referência da IAM API
- [Manual do usuário do AWS Certificate Manager](#)

Trabalhar com Kinesis

Esta seção fornece exemplos de programação [Amazon Kinesis](#) usando o AWS SDK for Java 2.x.

Para ter mais informações sobre o Kinesis, [consulte o Amazon Kinesis Guia do desenvolvedor](#).

Os exemplos a seguir incluem somente o código necessário para demonstrar cada técnica. O [código de exemplo completo está disponível em GitHub](#). A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

- [Inscrever-se no Amazon Kinesis Data Streams](#)

Inscrever-se no Amazon Kinesis Data Streams

Os exemplos a seguir mostram como recuperar e processar dados de fluxos de Amazon Kinesis dados usando o `subscribeToShard` método. Kinesis Data Streams agora emprega o recurso aprimorado de fanout e uma API de recuperação de dados HTTP/2 de baixa latência, facilitando aos desenvolvedores a execução de vários aplicativos de baixa latência e alto desempenho no mesmo fluxo de dados. Kinesis

Configurar

Primeiro, crie um Kinesis cliente assíncrono e um objeto. [SubscribeToShardRequest](#) Esses objetos são usados em cada um dos exemplos a seguir para se inscrever em eventos do Kinesis.

Importações

```
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;
```

Código

```
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 a interface do construtor

Você pode usar o `builder` método para simplificar a criação do [SubscribeToShardResponseHandler](#).

Usando o construtor, você pode definir cada retorno de chamada do ciclo de vida com um método de chamada em vez de implementar a interface completa.

Código

```
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);
}
```

Para ter mais controle do editor, você pode usar o método `publisherTransformer` para personalizar o editor.

Código

```
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);
}
```

Veja o [exemplo completo](#) em GitHub.

Use um manipulador de respostas personalizado

Para controle total do assinante e do editor, implemente a [`SubscribeToShardResponseHandler` interface](#).

Neste exemplo, você implementa o método `onEventStream`, que permite o acesso total ao editor. Isso demonstra como transformar o editor em registros de eventos para impressão pelo assinante.

Código

```
private static CompletableFuture<Void>
responseHandlerBuilderClassic(KinesisAsyncClient client, SubscribeToShardRequest
request) {
    SubscribeToShardResponseHandler responseHandler = new
SubscribeToShardResponseHandler() {
```

```
    @Override
    public void responseReceived(SubscribeToShardResponse response) {
        System.out.println("Received 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);
}
```

Veja o [exemplo completo](#) em GitHub.

Use a interface do visitante

Você pode usar um objeto [Visitante](#) para se inscrever em eventos específicos que tenha interesse em assistir.

Código

```
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);
}
```

Veja o [exemplo completo](#) em GitHub.

Use um assinante personalizado

Você também pode implementar seu próprio assinante personalizado para se inscrever no streaming.

Este trecho de código mostra um exemplo de assinante.

Código

```
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");
    }
}
```

Você pode passar o assinante personalizado para o `subscribe` método, conforme mostrado no trecho de código a seguir.

Código

```
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);
}
```

Veja o [exemplo completo em GitHub](#).

Gravar registros de dados em um fluxo Kinesis de dados

Você pode usar o [KinesisClient](#) objeto para gravar registros de dados em um fluxo de Kinesis dados usando o `putRecords` método. Para invocar esse método com êxito, crie um [PutRecordsRequest](#) objeto. Você passa o nome do fluxo de dados para o `streamName` método. Além disso, passe os dados usando o método `putRecords` (como mostrado no exemplo de código a seguir).

Importações

```
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;
```

No exemplo de código Java a seguir, observe que o `StockTrade` objeto é usado como dados para gravar no fluxo Kinesis de dados. Antes de executar esse exemplo, certifique-se de ter criado o fluxo de dados.

Código

```
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);  
}  
}
```

Veja o [exemplo completo](#) em GitHub.

Use uma biblioteca de terceiros

Você pode usar outras bibliotecas de terceiros em vez de implementar um assinante personalizado. Este exemplo demonstra o uso da RxJava implementação, mas você pode usar qualquer biblioteca que implemente as interfaces do Reactive Streams. Consulte a [página RxJava wiki no Github](#) para obter mais informações sobre essa biblioteca.

Para usar a biblioteca, adicione-a como uma dependência. Se estiver usando o Maven, o exemplo mostra o trecho POM a ser usado.

Entrada POM

```
<dependency>  
    <groupId>io.reactivex.rxjava2</groupId>  
    <artifactId>rxjava</artifactId>  
    <version>2.1.14</version>  
</dependency>
```

Importações

```
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;
```

Este exemplo é usado RxJava no método de `onEventStream` ciclo de vida. Assim, você tem acesso total ao editor, que pode ser usado para criar um Rx Flowable.

Código

```
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();
```

Você também pode usar o método `publisherTransformer` com o editor `Flowable`. Você deve adaptar o `Flowable` editor a um `SdkPublisher`, conforme mostrado no exemplo a seguir.

Código

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
```

```
.build();
```

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [SubscribeToShardEvent](#)na referência da Amazon Kinesis API
- [SubscribeToShard](#)na referência da Amazon Kinesis API

Invocar, listar e excluir funções AWS Lambda

Esta seção fornece exemplos de programação com o cliente Lambda de serviço usando o AWS SDK for Java 2.x.

Tópicos

- [Invocar uma função do Lambda](#)
- [Listar funções do Lambda](#)
- [Excluir uma função do Lambda](#)

Invocar uma função do Lambda

Você pode invocar uma Lambda função criando um [LambdaClient](#)objeto e invocando seu `invoke` método. Crie um [InvokeRequest](#)objeto para especificar informações adicionais, como o nome da função e a carga útil a ser passada para a Lambda função. Os nomes das funções aparecem como `arn:aws:lambda:us-east-1:123456789012:function::HelloFunction` Você pode recuperar o valor observando a função no AWS Management Console.

Para passar dados de carga para uma função, crie um [SdkBytes](#)objeto que contenha informações. Por exemplo, no exemplo de código a seguir, observe os dados JSON passados para a função do Lambda.

Importações

```
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;
```

Código

O exemplo de código a seguir demonstra como invocar uma função do Lambda.

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Listar funções do Lambda

Crie um objeto [LambdaClient](#) e invoque seu `listFunctions` método. Esse método retorna um [ListFunctionsResponse](#) objeto. Você pode invocar o `functions` método desse objeto para retornar uma lista de [FunctionConfiguration](#) objetos. É possível percorrer a lista para recuperar informações sobre as funções. Por exemplo, o exemplo de código Java a seguir mostra como obter cada nome de função.

Importações

```
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;
```

Código

O exemplo de código Java a seguir demonstra como recuperar uma lista de nomes de função.

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Excluir uma função do Lambda

Crie um [`LambdaClient`](#) objeto e invoque seu `deleteFunction` método. Crie um [`DeleteFunctionRequest`](#) objeto e passe-o para o `deleteFunction` método. Esse objeto contém informações como o nome da função a ser excluída. Os nomes das funções aparecem como `arn:aws:lambda:us-east-1:123456789012:function:HelloFunction`. Você pode recuperar o valor observando a função no AWS Management Console.

Importações

```
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;
```

Código

O código Java a seguir demonstra como excluir uma Lambda função.

```
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);
    }
}
```

Veja o [exemplo completo](#) emGitHub.

Trabalhar com Amazon Pinpoint

É possível usar o Amazon Pinpoint para enviar mensagens relevantes e personalizadas para seus clientes por meio de vários canais de comunicação, como notificações por push, SMS e e-mail.

Criar um projeto

Um projeto (ou aplicativo) no Amazon Pinpoint é um conjunto de configurações, dados de clientes, segmentos e campanhas.

Para criar um projeto, comece construindo um [CreateApplicationRequest](#) objeto com o nome do projeto como o valor de `name()`. Em seguida, construa um [CreateAppRequest](#) objeto, passando o `CreateApplicationRequest` objeto como o valor de seu `createApplicationRequest()` método. Chame o método `createApp()` do [PinpointClient](#) transmitindo o objeto `CreateAppRequest`. Capture o resultado dessa solicitação como um objeto [CreateAppResponse](#), conforme demonstrado no trecho de código a seguir.

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) em GitHub.

Criar um segmento dinâmico

Um segmento é um conjunto de clientes que compartilham atributos específicos, como a cidade em que vivem ou a frequência com que visitam seu site. Um segmento dinâmico é aquele que é baseado em atributos definidos por você e que pode mudar ao longo do tempo.

Para criar um segmento dinâmico, primeiro compile todas as dimensões desejadas para esse segmento. Por exemplo, o seguinte trecho de código está definido para incluir clientes

que estavam ativos no site nos últimos 30 dias. Você pode fazer isso criando primeiro um [RecencyDimension](#) objeto com o `duration()` e `recencyType()` desejado (ou seja, ou seja, ACTIVE ou INACTIVE) e depois passando esse objeto para um objeto [SegmentBehaviors](#) construtor como o valor `derecency()`.

Depois de definir os atributos do segmento, crie-os em um [SegmentDimensions](#) objeto. Em seguida, construa um [WriteSegmentRequest](#) objeto, passando o `SegmentDimensions` objeto como o valor de `seudimensions()`. Em seguida, construa um [CreateSegmentRequest](#) objeto, passando o `WriteSegmentRequest` objeto como o valor de `seuwriteSegmentRequest()`. Finalmente, chame o método `createSegment()` do `PinpointClient` transmitindo o objeto `CreateSegmentRequest`. Capture o resultado dessa solicitação como um [CreateSegmentResponse](#) objeto.

Importações

```
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;
```

Código

```
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;
}
```

Veja o [exemplo completo](#) em GitHub.

Importar um segmento estático

Um segmento estático é aquele que você cria e importa de fora do Amazon Pinpoint. O código de exemplo a seguir mostra como criar um segmento estático importando-o do Amazon S3.

Pré-requisito

Antes de concluir este exemplo, é necessário criar uma função do IAM que conceda ao Amazon Pinpoint acesso ao Amazon S3. Para obter mais informações, consulte a [IAMfunção para importar endpoints ou segmentos no Guia do Amazon Pinpoint desenvolvedor](#).

Para importar um segmento estático, comece criando um `ImportJobRequest` objeto. No compilador, especifique o `s3Url()`, o `roleArn()` e o `format()`.

Note

Para obter mais informações sobre as propriedades de um `ImportJobRequest`, consulte a [ImportJobRequest seção Importar trabalhos](#) na referência da Amazon Pinpoint API.

Em seguida, construa um `CreateImportJobRequest` `ImportJobRequest` objeto, passando o objeto como o valor dele e o ID do seu projeto como `applicationId()` ou `importJobRequest()`. Chame o método `createImportJob()` do `PinpointClient` transmitindo o objeto `CreateImportJobRequest`. Capture o resultado dessa solicitação como um objeto `CreateImportJobResponse`, conforme demonstrado no trecho de código a seguir.

Importações

```
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;
```

Código

```
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;
}
```

Veja o [exemplo completo](#) emGitHub.

Listar segmentos para o projeto

Para listar os segmentos associados a um projeto específico, comece construindo um [GetSegmentsRequest](#) objeto, com o ID do projeto como o valor do seu `applicationId()`.

Depois, chame o método `getSegments()` do `PinpointClient` transmitindo o objeto `GetSegmentsRequest`. Capture o resultado dessa solicitação como um [GetSegmentsResponse](#) objeto. Finalmente, instancie um objeto [List](#) enviado para a classe [SegmentResponse](#). Chame o `segmentsResponse().item()` de `GetSegmentsResponse`, como demonstrado no seguinte trecho de código. A partir daí, é possível iterar os resultados.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em [GitHub](#).

Criar uma campanha

Uma campanha é uma iniciativa destinada a envolver um determinado segmento de público enviando mensagens para esses clientes.

Para criar uma campanha, primeiro compile todas as configurações desejadas para esta campanha. No trecho de código a seguir, por exemplo, a campanha será iniciada imediatamente porque o `startTime()` da [Programação](#) está definido como `IMMEDIATE`. Para configurá-la para iniciar em um horário específico, especifique uma hora no formato ISO 8601.

 Note

Para obter mais informações sobre as configurações disponíveis para campanhas, consulte a seção Programação de [Campanhas](#) na Referência da Amazon Pinpoint API.

Depois de definir a configuração da sua campanha, transforme-a em um [WriteCampaignRequest](#) objeto. Nenhum dos métodos do `builder()` da [WriteCampaignRequest](#) são necessários. Mas você precisa incluir qualquer uma das configurações ([MessageConfiguration](#)) definidas para a campanha. Também recomendamos que você inclua um `name` e uma `description` para a campanha para que você possa distingui-la facilmente de outras campanhas. Chame o método `createCampaign()` do `PinpointClient` transmitindo o objeto [WriteCampaignRequest](#). Capture o resultado dessa solicitação como um [CreateCampaignResponse](#) objeto.

Importações

```
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;
```

Código

```
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;
}
```

Veja o [exemplo completo](#) emGitHub.

Enviar uma mensagem

Para enviar uma mensagem de texto SMSAmazon Pinpoint, primeiro crie um [AddressConfiguration](#) objeto para especificar `channelType()` o. (No exemplo a seguir, ele está definido como `ChannelType.SMS` para indicar que a mensagem será enviada via SMS.) Inicialize a [HashMap](#) para armazenar o número de telefone de destino e o [AddressConfiguration](#) objeto. Compile um objeto [SMSMessage](#) que contém os valores relevantes. Eles incluem o `originationNumber`, o tipo de mensagem (`messageType`) e o body da própria mensagem.

Depois de criar a mensagem, transforme o [SMSMessage](#) objeto em um [DirectMessageConfiguration](#) objeto. Construa seu objeto de [Mapa](#) e [DirectMessageConfiguration](#) objeto em um [MessageRequest](#) objeto. Crie um [SendMessagesRequest](#) objeto, incluindo seu ID de projeto (`applicationId`) e seu [MessageRequest](#) objeto. Chame o método `sendMessages()` do [PinpointClient](#) transmitindo o objeto [SendMessagesRequest](#). Capture o resultado dessa solicitação como um [SendMessagesResponse](#) objeto.

Importações

```
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;
```

Código

```
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);
}
}
```

Veja o [exemplo completo](#) em GitHub.

Para obter mais informações, consulte o [Guia do desenvolvedor do Amazon Pinpoint](#).

Trabalhe com o Amazon S3

Esta seção fornece exemplos de programação com [Amazon Simple Storage Service\(S3\)](#) usando o AWS SDK for Java 2.x

Os exemplos a seguir incluem somente o código necessário para demonstrar cada técnica. O [código de exemplo completo está disponível em GitHub](#). A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Note

A partir da versão 2.18.x, o AWS SDK for Java 2.x usa endereçamento no estilo de host virtual ao incluir uma substituição de endpoint. Isso se aplica desde que o nome do bucket seja um rótulo DNS válido.

Chame o [forcePathStyle](#)método true no seu construtor de clientes para forçar o cliente a usar o endereçamento no estilo de caminho para buckets.

O exemplo a seguir mostra um cliente de serviço configurado com uma substituição de endpoint e usando endereçamento no estilo de caminho.

```
S3Client client = S3Client.builder()
```

```
.region(Region.US_WEST_2)
.endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
.forcePathStyle(true)
.build();
```

Use pontos de acesso ou pontos de acesso multirregionais

Depois que os pontos de acesso ou [Pontos de acesso multirregionais do Amazon S3](#) forem configurados, você poderá chamar métodos de objeto, como `putObject` e, `getObject` e fornecer o identificador do ponto de acesso em vez do nome do bucket.

Por exemplo, se o identificador ARN do ponto de acesso for `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, você poderá usar o seguinte trecho para chamar o método `putObject`

```
Path path = Paths.get(URI.create("file:///temp/file.txt"));

s3Client.putObject(builder -> builder
    .key("myKey")
    .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")
, path);
```

No lugar da string ARN, você também pode usar o [alias em estilo de bucket](#) do ponto de acesso para o parâmetro `bucket`.

Para usar o ponto de acesso multirregional, substitua o bucket parâmetro pelo ARN do ponto de acesso multirregional que tenha o seguinte formato.

```
arn:aws:s3:::account-id:accesspoint/MultiRegionAccessPoint_alias
```

Adicione a seguinte dependência do Maven para trabalhar com pontos de acesso multirregionais usando o SDK for Java. Pesquise a [versão mais recente](#) no maven central.

```
<dependency>
<groupId>software.amazon.awssdk</groupId>
<artifactId>auth-crt</artifactId>
<version>VERSION</version>
```

```
</dependency>
```

Tópicos

- [Criar, listar e excluir Amazon S3 compartimentos](#)
- [Trabalhe com Amazon S3 objetos](#)
- [Trabalhe com Amazon S3 URLs pré-assinados](#)
- [Acesso entre regiões para o Amazon S3](#)
-

Criar, listar e excluir Amazon S3 compartimentos

Cada objeto (arquivo) no Amazon S3 deve residir em um bucket. Um bucket representa um conjunto (contêiner) de objetos. Cada bucket deve ter uma chave (nome) exclusiva. Para obter informações detalhadas sobre compartimentos e sua configuração, consulte [Trabalhando com Amazon S3 compartimentos](#) no Guia do Amazon Simple Storage Service usuário.

Note

Melhor prática

Recomendamos que você habilite a regra do [AbortIncompleteMultipartUpload](#) ciclo de vida em seus buckets. Amazon S3

Essa regra leva o Amazon S3 a anular multipart uploads que não sejam concluídos dentro de um número específico de dias depois de serem iniciados. Quando o limite de tempo definido é excedido, o Amazon S3 anula o upload e exclui os dados de uploads incompletos.

Para obter mais informações, consulte [Configuração do ciclo de vida de um bucket com controle de versão no Guia do usuário](#). Amazon Simple Storage Service

Note

Esses trechos de código pressupõem que você entenda o material no básico e tenha configurado as AWS credenciais padrão usando as informações em. [the section called “Configurar o acesso de login único para o SDK”](#)

Criar um bucket

Crie um [CreateBucketRequest](#) e forneça um nome de bucket. Passe-o para o método do S3Client. `createBucket` Use o S3Client para realizar operações adicionais, como listar ou excluir buckets, conforme mostrado nos exemplos mais adiante.

Importações

```
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;
```

Código

Primeiro, crie um S3Client.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Faça uma solicitação de criação de bucket.

```
// 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);
}
}
```

Veja o [exemplo completo](#) emGitHub.

Listar buckets

Construa um [ListBucketsRequest](#). Use o `listBuckets` método do `S3Client` para recuperar a lista de buckets. Se a solicitação for bem-sucedida, um [ListBucketsResponse](#) será retornado. Use este objeto de resposta para recuperar a lista de buckets.

Importações

```
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;
```

Código

Primeiro, crie um S3Client.

```
ProfileCredentialsProvider credentialsProvider =  
ProfileCredentialsProvider.create();  
Region region = Region.US_EAST_1;  
S3Client s3 = S3Client.builder()  
.region(region)  
.credentialsProvider(credentialsProvider)  
.build();
```

Faça uma solicitação de listagem de buckets.

```
// List buckets  
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();  
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);  
listBucketsResponse.buckets().stream().forEach(x ->  
System.out.println(x.name()));
```

Veja o [exemplo completo](#) em GitHub.

Excluir um bucket

Para que você possa excluir um bucket do Amazon S3, deverá verificar se o bucket está vazio, ou o serviço retornará um erro. Se você tiver um [bucket versionado](#), deverá também excluir todos os objetos versionados que estão no bucket.

Tópicos

- [Excluir objetos em um bucket](#)
- [Excluir um bucket vazio](#)

Excluir objetos em um bucket

Crie um [ListObjectsV2Request](#) e use o `listObjects` método do S3Client para recuperar a lista de objetos no bucket. Em seguida, use o método `deleteObject` em cada objeto para excluí-lo.

Importações

```
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;
```

Código

Primeiro, crie um S3Client.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Excluir todos os objetos no 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());
    } catch (S3Exception e) {
        System.out.println("An error occurred while deleting objects in the bucket " + bucket);
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        }
    } while (listObjectsV2Response.isTruncated());
```

Veja o [exemplo completo](#) em GitHub.

Excluir um bucket vazio

Crie um [DeleteBucketRequest](#) com um nome de bucket e passe-o para o método do S3Client.

```
deleteBucket
```

Importações

```
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;
```

Código

Primeiro, crie um S3Client.

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();
```

Excluir o bucket.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();
```

```
s3.deleteBucket(deleteBucketRequest);
s3.close();
```

Veja o [exemplo completo](#) em GitHub.

Trabalhe com Amazon S3 objetos

Um objeto do Amazon S3 representa um arquivo ou um conjunto de dados. Cada objeto deve estar contido em um [bucket](#).

Note

Melhor prática

Recomendamos que você habilite a regra do [AbortIncompleteMultipartUpload](#) ciclo de vida em seus buckets. Amazon S3

Essa regra leva o Amazon S3 a anular multipart uploads que não sejam concluídos dentro de um número específico de dias depois de serem iniciados. Quando o limite de tempo definido é excedido, o Amazon S3 anula o upload e exclui os dados de uploads incompletos.

Para obter mais informações, consulte [Configuração do ciclo de vida de um bucket com controle de versão no Guia do usuário](#). Amazon Simple Storage Service

Note

Esses trechos de código pressupõem que você entenda o material no básico e tenha configurado as AWS credenciais padrão usando as informações em. [the section called “Configurar o acesso de login único para o SDK”](#)

Tópicos

- [Fazer upload de um objeto](#)
- [Fazer upload de objetos em várias partes](#)
- [Excluir um objeto](#)
- [Copiar um objeto](#)
- [List objects](#)
- [Mais exemplos](#)

Fazer upload de um objeto

Crie um [PutObjectRequest](#) forneça um nome de bucket e um nome de chave. Em seguida, use o `putObject` método do `S3Client` com um [RequestBody](#) que contém o conteúdo do objeto e o `PutObjectRequest` objeto. O bucket deve existir, ou o serviço retornará um erro.

Importações

```
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;
```

Código

```
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)));
```

Veja o [exemplo completo](#) em GitHub.

Fazer upload de objetos em várias partes

Use o `createMultipartUpload` método do `S3Client` para obter um ID de upload. Em seguida, use o método `uploadPart` para fazer upload de cada parte. Finalmente, use o `completeMultipartUpload` método do `S3Client` para fazer a fusão Amazon S3 de todas as partes carregadas e concluir a operação de upload.

Importações

```
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;
```

Código

```
// 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);
```

Veja o [exemplo completo](#) em GitHub.

Excluir um objeto

Crie um [DeleteObjectRequest](#) forneça um nome de bucket e um nome de chave. Use o `deleteObject` método do `S3Client` e passe a ele o nome de um bucket e objeto a serem excluídos. O bucket e a chave de objeto especificados devem existir, ou o serviço retornará um erro.

Importações

```
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;
```

Código

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

s3.deleteObject(deleteObjectRequest);
```

Veja o [exemplo completo](#) em GitHub.

Copiar um objeto

Crie um [CopyObjectRequest](#) fornecendo um nome de bucket para o qual o objeto seja copiado, um valor de string codificado em URL (consulte o método URLEncoder.encode) e o nome da chave do objeto. Use o copyObject método do S3Client e passe o [CopyObjectRequest](#) objeto. O bucket e a chave de objeto especificados devem existir, ou o serviço retornará um erro.

Importações

```
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;
```

Código

```
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 "";  
}
```

Veja o [exemplo completo](#) em GitHub.

List objects

Crie um [ListObjectsRequest](#) forneça o nome do bucket. Em seguida, invoque o `listObjects` método do S3Client e passe o objeto. [ListObjectsRequest](#) Esse método retorna um [ListObjectsResponse](#) que contém todos os objetos no bucket. Você pode invocar o método `contents` desse objeto para obter uma lista de objetos. É possível percorrer essa lista para exibir os objetos, conforme mostrado no exemplo de código a seguir.

Importações

```
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;
```

Código

```
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;  
}
```

Veja o [exemplo completo](#) emGitHub.

Mais exemplos

A seção de [exemplos de código](#) deste guia contém mais exemplos de como trabalhar com objetos do Amazon S3, incluindo como [baixar um objeto](#).

Trabalhe com Amazon S3 URLs pré-assinados

Você pode usar um objeto [S3Presigner](#) para assinar um para Amazon S3 SdkRequest que ele seja executado sem exigir autenticação por parte do chamador. Por exemplo, vamos supor que Alice tenha acesso a um objeto do S3 e queira compartilhar temporariamente o acesso a esse objeto com Bob. Alice pode gerar um [GetObjectRequest](#) objeto pré-assinado para compartilhar com Bob para

que ele possa baixar o objeto sem precisar acessar as credenciais de Alice. Você pode gerar URLs pré-assinados para solicitações HTTP GET e HTTP PUT.

Gere um URL pré-assinado e faça o upload de um arquivo

O exemplo de código a seguir contém três métodos. O primeiro método, `createPresignedUrl`, gera uma URL assinada que você pode usar posteriormente em uma solicitação HTTP PUT. Os próximos dois métodos mostram opções que usam a URL para realizar um HTTP PUT com um arquivo para conteúdo.

O `createPresignedUrl` método aceita nomes de intervalos e chaves junto com um tipo de conteúdo e um mapa de metadados. A primeira etapa do método usa os argumentos para criar um `PutObjectRequest`. Em seguida, o código envolve o `PutObjectRequest` em `aPutObjectPresignRequest`, que é passado como parâmetro para o [S3Presigner's presignPutObject](#) método.

A URL assinada que é retornada do método é válida por 10 minutos. Agora, qualquer cliente HTTP pode usar essa URL com uma solicitação HTTP PUT.

Os `useHttpClientToPut` métodos `useHttpURLConnectionToPut` e mostram o uso de diferentes clientes HTTP para carregar o arquivo. Em ambos os casos, observe que o tipo de conteúdo e os metadados devem ser adicionados aos cabeçalhos da solicitação HTTP PUT para corresponder ao que foi adicionado ao original. `PutObjectRequest`

Importações

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
```

```
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Código

O exemplo de código Java a seguir mostra um método que cria uma URL assinada e dois métodos que você pode usar para realizar uma solicitação HTTP PUT com a URL.

```
/**
 * Create a presigned URL for uploading with a PUT request.
 * @param bucketName - The name of the bucket.
 * @param keyName - The name of the object.
 * @param contentType - The content type of the object.
 * @param metadata - The metadata to store with the object.
 * @return - The presigned URL for an HTTP PUT.
 */
public URL createPresignedUrl(String bucketName, String keyName, String
contentType, Map<String, String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

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

        PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL will expire
in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
```

```
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("Which HTTP method needs to be used when uploading a file:
[{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url();
    }

}

/***
 * Use the JDK HttpURLConnection (since v1.1) class to do the upload, but you can
 * use any HTTP client.
 *
 * @param presignedUrl - The presigned URL.
 * @param fileToPut    - The file to upload.
 * @param contentType - The content type of the file.
 * @param metadata     - The metadata to store with the object.
 */
public void useHttpURLConnectionToPut(URL presignedUrl, File fileToPut, String
contentType, Map<String, String> metadata) {
    logger.info("Begin [{}] upload", contentType);
    try {
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        connection.setRequestProperty("Content-Type", contentType);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" + k,
v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
}
```

```
        }

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

/***
 * Use the JDK HttpClient (since v11) class to do the upload, but you can
 * use any HTTP client.
 *
 * @param presignedUrl - The presigned URL.
 * @param fileToPut    - The file to upload.
 * @param contentType - The content type of the file.
 * @param metadata     - The metadata to store with the object.
 */
public void useHttpClientToPut(URL presignedUrl, File fileToPut, String
contentType, Map<String, String> metadata) {
    logger.info("Begin [{}] upload", contentType);

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .header("Content-Type", contentType)

.PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
            .build(),
        HttpResponse.BodyHandlers.discard());
    }

    logger.info("HTTP response code is " + response.statusCode());

} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
}
```

Veja o [exemplo completo](#) em GitHub.

O exemplo anterior mostra uma solicitação PUT que usa um arquivo como conteúdo. A seção Exemplos de código deste guia mostra um [exemplo semelhante que carrega uma string](#) em vez de um arquivo.

Obter um objeto pré-assinado

Crie um objeto [S3Presigner](#) que represente o objeto cliente. Em seguida, crie um [GetObjectRequest](#) objeto e especifique o nome do bucket e o nome da chave. Além disso, crie um [GetObjectPresignRequest](#) objeto que possa ser executado posteriormente sem precisar de assinatura ou autenticação adicional. Ao criar esse objeto, você pode especificar o tempo em minutos que o bucket pode ser acessado sem usar credenciais invocando o método `signatureDuration` (conforme mostrado no exemplo de código a seguir).

Invoca o `presignGetObject` método que pertence ao objeto [S3Presigner](#) para criar um objeto [PresignedGetObjectRequest](#). Você pode invocar o método `url` desse objeto para obter a URL a ser usada. Quando tiver a URL, você poderá usar a lógica Java HTTP padrão para ler o conteúdo do bucket, conforme mostrado no exemplo de código Java a seguir.

Importações

```
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;
```

Código

O exemplo de código Java a seguir lê o conteúdo de um bucket do S3 pré-assinado.

```
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.printStackTrace();
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Acesso entre regiões para o Amazon S3

Quando você trabalha com buckets do Amazon Simple Storage Service (Amazon S3), você geralmente sabe o Região da AWS para o balde. A região com a qual você trabalha é determinada quando você cria o cliente S3.

No entanto, às vezes você pode precisar trabalhar com um bucket específico, mas não sabe se ele está localizado na mesma região definida para o cliente S3.

Em vez de fazer mais chamadas para determinar a região do bucket, você pode usar o SDK para permitir o acesso aos buckets do S3 em diferentes regiões.

Configuração

O suporte para acesso entre regiões tornou-se disponível com a versão 2.20.111 do SDK. Use esta versão ou uma versão posterior em seu arquivo de compilação do Maven para os dependências, conforme mostrado no trecho a seguir.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
    <version>2.20.111</version>
</dependency>
```

Em seguida, ao criar seu cliente S3, ative o acesso entre regiões conforme mostrado no trecho. Por padrão, o acesso não está habilitado.

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
    .build();
```

Como o SDK fornece acesso entre regiões

Quando você faz referência a um bucket existente em uma solicitação, como quando você usa `putObject` método, o SDK inicia uma solicitação para a região configurada para o cliente.

Se o bucket não existir nessa região específica, a resposta de erro incluirá a região real em que o bucket reside. Em seguida, o SDK usa a região correta em uma segunda solicitação.

Para otimizar solicitações futuras para o mesmo bucket, o SDK armazena em cache esse mapeamento de região no cliente.

Considerações

Ao habilitar o acesso ao bucket entre regiões, saiba que a primeira chamada de API pode resultar em maior latência se o bucket não estiver na região configurada do cliente. No entanto, as chamadas subsequentes se beneficiam das informações da região em cache, resultando em melhor desempenho.

Quando você ativa o acesso entre regiões, o acesso ao bucket não é afetado. O usuário deve estar autorizado a acessar o bucket em qualquer região em que ele resida.

O Amazon Simple Storage Service (Amazon S3) oferece a capacidade de especificar uma soma de verificação quando você carrega um objeto. Quando você especifica uma soma de verificação, ela é armazenada com o objeto e pode ser validada quando o objeto é baixado.

As somas de verificação fornecem uma camada adicional de integridade de dados quando você transfere arquivos. Com somas de verificação, você pode verificar a consistência dos dados confirmando que o arquivo recebido corresponde ao arquivo original. Para obter mais informações sobre somas de verificação com o Amazon S3, consulte [o Guia do usuário do Amazon Simple Storage Service](#).

Atualmente, o Amazon S3 oferece suporte a quatro algoritmos de soma de verificação: SHA-1, SHA-256, CRC-32 e CRC-32C. Você tem a flexibilidade de escolher o algoritmo mais adequado às suas necessidades e deixar que o SDK calcule a soma de verificação. Como alternativa, você pode especificar seu próprio valor de soma de verificação pré-computado usando um dos quatro algoritmos compatíveis.

Discutimos somas de verificação em duas fases de solicitação: upload de um objeto e download de um objeto.

Fazer upload de um objeto

Os valores válidos para o algoritmo são CRC32 CRC32CSHA1,, SHA256 e.

O trecho de código a seguir mostra uma solicitação para carregar um objeto com uma soma de verificação CRC-32. Quando o SDK envia a solicitação, ele calcula a soma de verificação CRC-32 e carrega o objeto. O Amazon S3 armazena a soma de verificação com o objeto.

Se a soma de verificação calculada pelo SDK não corresponder à soma de verificação calculada pelo Amazon S3 ao receber a solicitação, um erro será retornado.

Use um valor de soma de verificação pré-calculado

Um valor de soma de verificação pré-calculado fornecido com a solicitação desativa a computação automática pelo SDK e, em vez disso, usa o valor fornecido.

O exemplo a seguir mostra uma solicitação com uma soma de verificação SHA-256 pré-calculada.

Se o Amazon S3 determinar que o valor da soma de verificação está incorreto para o algoritmo especificado, o serviço retornará uma resposta de erro.

Uploads de várias partes

Você também pode usar somas de verificação com uploads de várias partes.

Fazer download de um objeto

Quando você usa o método [getObject](#) para baixar um objeto, o SDK valida automaticamente a soma de verificação . enabled

A solicitação no trecho a seguir direciona o SDK a validar a soma de verificação na resposta calculando a soma de verificação e comparando os valores.

Se o objeto não tiver sido carregado com uma soma de verificação, nenhuma validação ocorrerá.

Um objeto no Amazon S3 pode ter várias somas de verificação, mas somente uma soma de verificação é validada no download. A precedência a seguir, com base na eficiência do algoritmo de soma de verificação, determina qual soma de verificação o SDK valida:

1. CRC-32C
2. CRC-32
3. SHA-1

4. SHA-256

Por exemplo, se uma resposta contiver somas de verificação CRC-32 e SHA-256, somente a soma de verificação CRC-32 será validada.

Trabalhar com Amazon Simple Notification Service

Com o Amazon Simple Notification Service, é possível enviar facilmente mensagens de notificação em tempo real dos aplicativos aos assinantes em vários canais de comunicação. Este tópico descreve como executar algumas das funções básicas do Amazon SNS.

Criar um tópico

Um tópico é um agrupamento lógico de canais de comunicação que define para quais sistemas enviar uma mensagem, por exemplo, espalhando uma mensagem do AWS Lambda e um webhook HTTP. Você envia mensagens para o Amazon SNS, e elas são distribuídas para os canais definidos no tópico. Isso disponibiliza as mensagens para os assinantes.

Para criar um tópico, primeiro crie um [CreateTopicRequest](#) objeto, com o nome do tópico definido usando o `name()` método no construtor. Envie o objeto de solicitação para o Amazon SNS usando o método `createTopic()` do [SnsClient](#). Você pode capturar o resultado dessa solicitação como um [CreateTopicResponse](#) objeto, conforme demonstrado no trecho de código a seguir.

Importações

```
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;
```

Código

```
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 "";
}
```

Veja o [exemplo completo](#) em GitHub.

Liste seus Amazon SNS tópicos

Para recuperar uma lista de seus Amazon SNS tópicos existentes, crie um [ListTopicsRequest](#) objeto. Envie o objeto de solicitação para o Amazon SNS usando o método `listTopics()` do `SnsClient`. Você pode capturar o resultado dessa solicitação como um [ListTopicsResponse](#) objeto.

O trecho de código a seguir imprime o código de status HTTP da solicitação e uma lista de nomes de recurso da Amazon (ARNs) para os tópicos do Amazon SNS.

Importações

```
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;
```

Código

```
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);
        }
    }
```

Veja o [exemplo completo](#) em GitHub.

Inscrever um endpoint em um tópico

Depois de criar um tópico, é possível configurar quais canais de comunicação serão endpoints para esse tópico. As mensagens são distribuídas para esses endpoints após o Amazon SNS recebê-las.

Para configurar um canal de comunicação como um endpoint para um tópico, inscreva esse endpoint no tópico. Para começar, construa um [SubscribeRequest](#) objeto. Especifique o canal de comunicação (por exemplo, lambda ou email) como protocol() o. Defina o endpoint() para o local de saída relevante (por exemplo, o ARN de uma Lambda função ou endereço de e-mail) e, em seguida, defina o ARN do tópico no qual você deseja se inscrever como o. topicArn() Envie o objeto de solicitação para Amazon SNS usando o subscribe() método doSnsClient. Você pode capturar o resultado dessa solicitação como um [SubscribeResponse](#) objeto.

O trecho de código a seguir mostra como inscrever um endereço de e-mail em um tópico.

Importações

```
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;
```

Código

```
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);
    }
}
```

Veja o [exemplo completo](#) em GitHub.

Publicar uma mensagem em um tópico

Depois de ter um tópico e um ou mais endpoints configurados para ele, será possível publicar uma mensagem nele. Para começar, construa um [PublishRequest](#) objeto. Especifique a `message()` a ser enviada e o ARN do tópico (`topicArn()`) para o qual enviá-la. Envie o objeto de solicitação para o Amazon SNS usando o método `publish()` do `SnsClient`. Você pode capturar o resultado dessa solicitação como um [PublishResponse](#) objeto.

Importações

```
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;
```

Código

```
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);
        }
    }
```

Veja o [exemplo completo](#) em GitHub.

Cancelar a inscrição de um endpoint de um tópico

Você pode remover os canais de comunicação configurados como endpoints de um tópico. Depois de fazer isso, o tópico em si continua a existir e distribuir mensagens para quaisquer endpoints finais configurados para esse tópico.

Para remover um canal de comunicação como um endpoint de um tópico, cancele a inscrição desse endpoint do tópico. Para começar, crie um [UnsubscribeRequest](#) objeto e defina o ARN do tópico do qual você deseja cancelar a assinatura como o `subscriptionArn()`. Envie o objeto de solicitação para o SNS usando o método `unsubscribe()` do `SnsClient`. Você pode capturar o resultado dessa solicitação como um [UnsubscribeResponse](#) objeto.

Importações

```
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;
```

Código

```
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);  
}  
}
```

Veja o [exemplo completo](#) em GitHub.

Excluir um tópico

Para excluir um Amazon SNS tópico, primeiro crie um [DeleteTopicRequest](#) objeto com o ARN do tópico definido como o `topicArn()` método no construtor. Envie o objeto de solicitação para o Amazon SNS usando o método `deleteTopic()` do `SnsClient`. Você pode capturar o resultado dessa solicitação como um [DeleteTopicResponse](#) objeto, conforme demonstrado no trecho de código a seguir.

Importações

```
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;
```

Código

```
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);  
    }  
}
```

```
    }  
}
```

Veja o [exemplo completo](#) emGitHub.

Para obter mais informações, consulte o [Guia do desenvolvedor do Amazon Simple Notification Service](#).

Trabalhar com Amazon Simple Queue Service

Esta seção fornece exemplos de programação [Amazon Simple Queue Service](#) usando o AWS SDK for Java 2.x.

Os exemplos a seguir incluem somente o código necessário para demonstrar cada técnica. O [código de exemplo completo está disponível em GitHub](#). A partir daí, você pode fazer download de um único arquivo de origem ou clonar o repositório de maneira local para obter todos os exemplos para compilação e execução.

Tópicos

- [Trabalhe comAmazon Simple Queue Servicefilas de mensagens](#)
- [Enviar, receber e excluirAmazon Simple Servicemensagens](#)

Trabalhe comAmazon Simple Queue Servicefilas de mensagens

Uma fila de mensagens é o contêiner lógico usado para enviar mensagens de maneira confiável no Amazon Simple Queue Service. Existem dois tipos de filas: padrão e First-In, First-Out (FIFO – Primeiro a entrar, primeiro a sair). Para saber mais sobre filas e as diferenças entre esses tipos, consulte a[Amazon Simple Queue ServiceGuia do desenvolvedor](#).

Este tópico descreve como criar, listar, excluir e obter o URL de uma fila do Amazon Simple Queue Service usando o AWS SDK for Java.

O `SqsClient`A variável usada nos exemplos a seguir pode ser criada a partir do seguinte trecho.

```
SqsClient sqsClient = SqsClient.create();
```

Quando você cria um`SqsClient`usando a estática`create()`método, a região carregada pelo[cadeia de fornecedores da região padrão](#)e as credenciais são carregadas pelo[cadeia de provedores de credenciais padrão](#).

Criar uma fila

Use o `SqsClient's createQueue` método e forneça um [CreateQueueRequest](#) objeto que descreve os parâmetros da fila.

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
    .queueName(queueName)
    .build();

sqcClient.createQueue(createQueueRequest);
```

Veja o [amostra completa](#) em GitHub.

Listar filas

Para listar o Amazon Simple Queue Service filas para sua conta, ligue para o `SqsClient's listQueues` método com um [ListQueuesRequest](#) objeto.

Usar a sobrecarga `listQueues` sem parâmetros retorna todas as filas, até 1.000 filas. Você pode fornecer um prefixo de nome da fila para o objeto `ListQueuesRequest` a fim de limitar os resultados para filas que correspondem a esse prefixo.

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
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);
}
```

Veja o [amostra completa](#) em GitHub.

Obter o URL de uma fila

Ligue para o SqsClient's getQueueUrl método. com um [GetQueueUrlRequest](#) objeto.

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
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 "";
```

Veja o [amostra completa](#) em GitHub.

Excluir uma fila

Forneça a fila[URL](#)para o[DeleteMessageRequest](#)objeto. Depois, chame o método `deleteQueue` do `SqsClient`.

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
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);
    }
}
```

Veja o[amostra completa](#)emGitHub.

Mais informações

- [ComoAmazon Simple Queue ServiceAs filas funcionam](#)[naAmazon Simple Queue Service](#)Guia do desenvolvedor

- [CreateQueue](#)na Amazon Simple Queue ServiceReferência da API
- [GetQueueUrl](#)na Amazon Simple Queue ServiceReferência da API
- [ListQueues](#)na Amazon Simple Queue ServiceReferência da API
- [DeleteQueues](#)na Amazon Simple Queue ServiceReferência da API

Enviar, receber e excluirAmazon Simple Queue Servicemensagens

Uma mensagem é um trecho de dados que pode ser enviado e recebido por componentes distribuídos. As mensagens são sempre entregues usando-se uma [fila do SQS](#).

O `SqsClient` variável usada nos exemplos a seguir pode ser criada a partir do seguinte trecho.

```
SqsClient sqsClient = SqsClient.create();
```

Quando você cria um `SqsClient` usando a estática `create()` método, a região é carregada pelo [cadeia de fornecedores da região padrão](#) e as credenciais são carregadas pelo [cadeia de provedores de credenciais padrão](#).

Enviar uma mensagem

Adicione uma mensagem única a uma fila do Amazon Simple Queue Service chamando o método `SqsClient` de cliente do `sendMessage`. Forneça um [SendMessageRequest](#) objeto que contém a fila [URL](#), corpo da mensagem e valor de atraso opcional (em segundos).

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
sqsClient.sendMessage(SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("Hello world!")
    .delaySeconds(10)
    .build());
```

Enviar várias mensagens em uma solicitação

Envie mais de uma mensagem em uma única solicitação usando o `SqsClient sendMessageBatch` método. Esse método requer um [SendMessageBatchRequest](#) que contém o URL da fila e uma lista de mensagens a serem enviadas. (Cada mensagem é uma [SendMessageBatchRequestEntry](#).) Você também pode atrasar o envio de uma mensagem específica, configurando um valor de atraso na mensagem.

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
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();
sqcClient.sendMessageBatch(sendMessageBatchRequest);
```

Veja o [amostra completa](#) em GitHub.

Recuperar mensagens

Recupere todas as mensagens que estão atualmente na fila chamando o `SqsClient receiveMessage` método. Esse método requer um [ReceiveMessageRequest](#) que contém o URL da fila. Você também pode especificar o número máximo de mensagens para retornar. As mensagens são retornadas como uma lista de objetos [Message](#).

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .maxNumberOfMessages(5)
    .build();
    List<Message> messages =
sqcClient.receiveMessage(receiveMessageRequest).messages();
    return messages;
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

Excluir uma mensagem após o recebimento

Depois de receber uma mensagem e processar seu conteúdo, exclua a mensagem da fila enviando o identificador de recebimento da mensagem e o URL da fila para o SqsClient deleteMessage método.

Importações

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

Código

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
    .queueUrl(queueUrl)
    .receiptHandle(message.receiptHandle())
    .build();
    sqcClient.deleteMessage(deleteMessageRequest);
```

}

Veja o [amostra completa](#) em GitHub.

Mais informações

- [Como Amazon Simple Queue Service As filas funcionam](#) na Amazon Simple Queue Service Guia do desenvolvedor
- [Send Message](#) na Amazon Simple Queue Service Referência da API
- [SendMessageBatch](#) na Amazon Simple Queue Service Referência da API
- [ReceiveMessage](#) na Amazon Simple Queue Service Referência da API
- [DeleteMessage](#) na Amazon Simple Queue Service Referência da API

Trabalhar com Amazon Transcribe

O exemplo a seguir mostra como o streaming bidirecional funciona usando o Amazon Transcribe. O streaming bidirecional indica que há um streaming de dados que vai para o serviço e que é recebido de volta em tempo real. O exemplo usa o transcrição de streaming do Amazon Transcribe para enviar um streaming de áudio e receber um streaming de texto transcreto em tempo real.

Consulte [Transcrição de streaming](#) no Guia do Amazon Transcribe desenvolvedor para saber mais sobre esse recurso.

Consulte [Primeiros passos](#) no Guia do Amazon Transcribe desenvolvedor para começar a usar Amazon Transcribe.

Configurar o microfone

Esse código usa o pacote javax.sound.sampled para fazer streaming de áudio de um dispositivo de entrada.

Código

```
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;
}
```

Veja o [exemplo completo](#) emGitHub.

Crie um editor

Esse código implementa um editor que publica dados de áudio do streaming de áudio do Amazon Transcribe.

Código

```
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 SubscriptionImpl(Subscriber<? super AudioStream> subscriber,
                           InputStream inputStream) {
        this.subscriber = subscriber;
        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);
            }
        });
    }

    private AudioEvent audioEventFromBuffer(ByteBuffer buffer) {
        // Implementation details...
    }
}
```

```
        } 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();
}
}
```

Veja o [exemplo completo](#) em GitHub.

Crie o cliente e inicie a transmissão

No método principal, crie um objeto de solicitação, inicie o streaming da entrada de áudio e instancie o editor com a entrada de áudio.

Você também deve criar um [StartStreamTranscriptionResponseHandler](#) para especificar como lidar com a resposta de Amazon Transcribe.

Depois, use o método `startStreamTranscription` do `TranscribeStreamingAsyncClient` para iniciar o streaming bidirecional.

Importações

```
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.StartStreamTranscriptionResponseHand
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

Código

```
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);  
}  
}
```

Veja o [exemplo completo](#) em GitHub.

Mais informações

- [Como funciona](#) no Guia do Amazon Transcribe Desenvolvedor.
- [Introdução ao streaming de áudio](#) no Guia do Amazon Transcribe desenvolvedor.
- [Diretrizes e limites](#) no Guia do Amazon Transcribe desenvolvedor.

Exemplos de código do SDK for Java 2.x

Os exemplos de código neste tópico mostram como usar o AWS SDK for Java 2.x withAWS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Exemplos entre serviços são amostras de aplicações que funcionam em vários Serviços da AWS.

Exemplos

- [Ações e cenários usando o SDK for Java 2.x](#)
- [Exemplos de serviços cruzados usando o SDK for Java 2.x](#)

Ações e cenários usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x withServiços da AWS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Serviços

- [Exemplos de API Gateway usando o SDK for Java 2.x](#)
- [Exemplos de controladores de recuperação de aplicativos usando o SDK for Java 2.x](#)
- [Exemplos do Aurora usando o SDK for Java 2.x](#)
- [Exemplos de Auto Scaling usando o SDK for Java 2.x](#)
- [CloudFront exemplos usando o SDK for Java 2.x](#)

- [CloudWatch exemplos usando o SDK for Java 2.x](#)
- [CloudWatch Exemplos de eventos usando o SDK for Java 2.x](#)
- [CloudWatch Exemplos de registros usando o SDK for Java 2.x](#)
- [Exemplos de provedores de identidade do Amazon Cognito usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Comprehend usando o SDK for Java 2.x](#)
- [Exemplos do DynamoDB usando o SDK for Java 2.x](#)
- [Exemplos do Amazon EC2 usando SDK for Java 2.x](#)
- [Exemplos do Amazon ECS usando SDK for Java 2.x](#)
- [Exemplos de Elastic Load Balancing usando o SDK for Java 2.x](#)
- [OpenSearch Exemplos de serviços usando o SDK for Java 2.x](#)
- [EventBridge exemplos usando o SDK for Java 2.x](#)
- [Exemplos de previsão usando o SDK for Java 2.x](#)
- [AWS Glueexemplos usando o SDK for Java 2.x](#)
- [HealthImaging exemplos usando o SDK for Java 2.x](#)
- [Exemplos de IAM usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Keyspaces usando o SDK for Java 2.x](#)
- [Exemplos do Kinesis usando o SDK for Java 2.x](#)
- [AWS KMSexemplos usando o SDK for Java 2.x](#)
- [Exemplos de Lambda usando SDK for Java 2.x](#)
- [MediaConvert exemplos usando o SDK for Java 2.x](#)
- [Exemplos do Migration Hub usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Personalize usando o SDK for Java 2.x](#)
- [Exemplos de eventos do Amazon Personalize usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Personalize Runtime usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Pinpoint usando o SDK for Java 2.x](#)
- [Exemplos de API de voz e SMS do Amazon Pinpoint usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Polly usando o SDK for Java 2.x](#)
- [Exemplos do Amazon RDS usando o SDK for Java 2.x](#)

- [Exemplos do Amazon Redshift usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Rekognition usando o SDK for Java 2.x](#)
- [Exemplos de registro de domínio do Route 53 usando o SDK for Java 2.x](#)
- [Exemplos do Amazon S3 usando SDK for Java 2.x](#)
- [Exemplos do S3 Glacier usando o SDK for Java 2.x](#)
- [SageMaker exemplos usando o SDK for Java 2.x](#)
- [Exemplos do Secrets Manager usando o SDK for Java 2.x](#)
- [Exemplos do Amazon SES usando SDK for Java 2.x](#)
- [Exemplos da API v2 do Amazon SES usando SDK for Java 2.x](#)
- [Exemplos do Amazon SNS usando SDK for Java 2.x](#)
- [Exemplos do Amazon SQS usando SDK for Java 2.x](#)
- [Exemplos de Step Functions usando o SDK for Java 2.x](#)
- [AWS STS exemplos usando o SDK for Java 2.x](#)
- [AWS Support exemplos usando o SDK for Java 2.x](#)
- [Exemplos do Systems Manager usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Textract usando o SDK for Java 2.x](#)

Exemplos de API Gateway usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with API Gateway.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Crie uma API REST

O exemplo de código a seguir mostra como criar uma API REST do API Gateway.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateRestApi](#) Referência AWS SDK for Java 2.x da API.

Excluir uma API REST

O exemplo de código a seguir mostra como excluir uma API REST do API Gateway.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteRestApi](#) Referência AWS SDK for Java 2.x da API.

Excluir uma implantação

O exemplo de código a seguir mostra como excluir uma implantação.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteDeployment](#) na Referência AWS SDK for Java 2.x da API.

Implantar uma API REST

O exemplo de código a seguir mostra como implantar uma API REST do API Gateway.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "" ;  
}
```

- Para obter detalhes da API, consulte [CreateDeployment](#) na Referência AWS SDK for Java 2.x da API.

Exemplos de controladores de recuperação de aplicativos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Application Recovery Controller.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Obtenha o estado de um controle de roteamento

O exemplo de código a seguir mostra como obter o estado de um controle de roteamento do Application Recovery Controller.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

- Para obter detalhes da API, consulte [GetRoutingControlState](#) Referência AWS SDK for Java 2.x da API.

Atualizar o estado de um controle de roteamento

O exemplo de código a seguir mostra como atualizar o estado de um controle de roteamento do Application Recovery Controller.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

- Para obter detalhes da API, consulte [UpdateRoutingControlState](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Aurora usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Aurora.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um cluster de banco de dados

O exemplo de código a seguir mostra como criar um cluster de banco de dados Aurora.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateDBCluster](#) na Referência de API do AWS SDK for Java 2.x.

Criar um parameter group de cluster de banco de dados

O exemplo de código a seguir mostra como criar um grupo de parâmetros de cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateDB ClusterParameterGroup na Referência AWS SDK for Java 2.x](#) da API.

Criar um snapshot de cluster de banco de dados

O exemplo de código a seguir mostra como criar um snapshot do cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateDB ClusterSnapshot na Referência AWS SDK for Java 2.x](#) da API.

Criar uma instância de banco de dados em um cluster de banco de dados

O exemplo de código a seguir mostra como criar uma instância de banco de dados em um cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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().dbInstanceState());
        return response.dbInstance().dbInstanceArn();

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

- Para ter detalhes da API, consulte [CreateDBInstance](#) na Referência de API do AWS SDK for Java 2.x.

Excluir um cluster de banco de dados

O exemplo de código a seguir mostra como excluir um cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para ter detalhes da API, consulte [DeleteDBCluster](#) na Referência de API do AWS SDK for Java 2.x.

Excluir um grupo de parâmetros de cluster de banco de dados

O exemplo de código a seguir mostra como excluir um grupo de parâmetros do cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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();
            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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteDB ClusterParameterGroup](#) na AWS SDK for Java 2.x Referência da API.

Excluir uma instância de banco de dados

O exemplo de código a seguir mostra como excluir uma instância de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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().dbInstanceState());
```

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

- Para ter detalhes da API, consulte [DeleteDBInstance](#) na Referência de API do AWS SDK for Java 2.x.

Descrever os grupos de parâmetros de cluster de banco de dados

O exemplo de código a seguir mostra como descrever os grupos de parâmetros do cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}

}
```

- Para obter detalhes da API, consulte [DescribeDB ClusterParameterGroups em Referência AWS SDK for Java 2.x](#) da API.

Descrever snapshots de cluster de banco de dados

O exemplo de código a seguir mostra como descrever os snapshots do cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DescribeDB ClusterSnapshots em Referência AWS SDK for Java 2.x da API](#).

Descrever clusters de banco de dados

O exemplo de código a seguir mostra como descrever os clusters de banco de dados Aurora.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para ter detalhes da API, consulte [DescribeDBClusters](#) na Referência de API do AWS SDK for Java 2.x.

Descrever instâncias de banco de dados

O exemplo de código a seguir mostra como descrever as instâncias de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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);
    }
}
```

- Para ter detalhes da API, consulte [DescribeDBInstances](#) na Referência de API do AWS SDK for Java 2.x.

Descrever as versões de mecanismo de banco de dados

O exemplo de código a seguir mostra como descrever as versões do mecanismo de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "+engine0b.dbParameterGroupFamily());  
        System.out.println("The name of the database engine  
"+engine0b.engine());  
        System.out.println("The version number of the database engine  
"+engine0b.engineVersion());  
    }  
  
} catch (RdsException e) {  
    System.out.println(e.getLocalizedMessage());  
    System.exit(1);  
}  
}
```

- Para obter detalhes da API, consulte [DescribeDB EngineVersions em Referência AWS SDK for Java 2.x](#) da API.

Descrever as opções de instâncias de banco de dados

O exemplo de código a seguir mostra como descrever as opções para instâncias de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

- Para obter detalhes da API, consulte [DescribeOrderableDB InstanceOptions](#) em Referência de AWS SDK for Java 2.x API.

Descrever os parâmetros de um grupo de parâmetros de cluster de banco de dados

O exemplo de código a seguir mostra como descrever parâmetros de um grupo de parâmetros do cluster de banco de dados Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeDB ClusterParameters em Referência AWS SDK for Java 2.x](#) da API.

Atualizar os parâmetros em um grupo de parâmetros de cluster de banco de dados

O exemplo de código a seguir mostra como atualizar parâmetros em um grupo de parâmetros do cluster de banco de dados Aurora.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ModifyDB ClusterParameterGroup na Referência AWS SDK for Java 2.x](#) da API.

Cenários

Começar a usar clusters de banco de dados

O código de exemplo a seguir mostra como:

- Crie um grupo de parâmetros de cluster do banco de dados do Aurora e defina os valores dos parâmetros.
- Crie um cluster de banco de dados que use o grupo de parâmetros.
- Crie uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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 requires an AWS Secrets Manager secret that contains the database  
 * credentials. If you do not create a  
 * secret, this example will not work. For details, see:  
 *  
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.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><secretName>" +  
            "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" +
```

```
"      secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\"\\n\" ;;

if (args.length != 7) {
    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 secretName = args[6];

// Retrieve the database credentials using AWS Secrets Manager.
Gson gson = new Gson();
User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
String username = user.getUsername();
String userPassword = user.getPassword();

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();
    }
```

```
private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

private static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();

    GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
    return valueResponse.secretString();
}

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();
            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;
                }
            }
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        throw new RuntimeException("Interrupted while deleting DB cluster group");
    }
}
```

```
        }
        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().dbInstanceState());}

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

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

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateDBCluster](#)
 - [Criado B ClusterParameterGroup](#)
 - [Criado B ClusterSnapshot](#)
 - [CreateDBInstance](#)
 - [DeleteDBCluster](#)
 - [Banco de dados excluído ClusterParameterGroup](#)
 - [DeleteDBInstance](#)
 - [DB descrito ClusterParameterGroups](#)
 - [DB descrito ClusterParameters](#)
 - [DB descrito ClusterSnapshots](#)
 - [DescribeDBClusters](#)
 - [DB descrito EngineVersions](#)
 - [DescribeDBInstances](#)
 - [DescribeOrderableDB InstanceOptions](#)
 - [Modificar banco de dados ClusterParameterGroup](#)

Exemplos de Auto Scaling usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um grupo

O exemplo de código a seguir mostra como criar um grupo de Auto Scaling.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                         String groupName,
                                         String launchTemplateName,
                                         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)
            .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);
    }
}
```

- Para obter detalhes da API, consulte [CreateAutoScalingGroup](#) a Referência AWS SDK for Java 2.x da API.

Excluir um grupo

O exemplo de código a seguir mostra como excluir um grupo do Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
}  
}
```

- Para obter detalhes da API, consulte [DeleteAutoScalingGroup](#) na Referência AWS SDK for Java 2.x da API.

Desativar a coleta de métricas para um grupo

O exemplo de código a seguir mostra como desativar a coleta de CloudWatch métricas para um grupo de Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DisableMetricsCollection](#) na Referência AWS SDK for Java 2.x da API.

Habilitar a coleta de métricas para um grupo

O exemplo de código a seguir mostra como ativar a coleta de CloudWatch métricas para um grupo de Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
        }
    }
```

- Para obter detalhes da API, consulte [EnableMetricsCollection](#) na Referência AWS SDK for Java 2.x da API.

Obtenha informações sobre grupos

O exemplo de código a seguir mostra como obter informações sobre grupos de Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [DescribeAutoScalingGroups](#) Referência AWS SDK for Java 2.x da API.

Obtenha informações sobre instâncias

O exemplo de código a seguir mostra como obter informações sobre instâncias do Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeAutoScalingInstances](#) a Referência AWS SDK for Java 2.x da API.

Obtenha informações sobre atividades de escalabilidade

O exemplo de código a seguir mostra como obter informações sobre as atividades do Auto Scaling.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DescribeScalingActivities](#) Referência AWS SDK for Java 2.x da API.

Defina a capacidade desejada de um grupo

O exemplo de código a seguir mostra como definir a capacidade desejada de um grupo de Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [SetDesiredCapacity](#) a Referência AWS SDK for Java 2.x da API.

Encerrar uma instância em um grupo

O exemplo de código a seguir mostra como encerrar uma instância em um grupo do Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [TerminateInstanceInAutoScalingGroup](#) a Referência AWS SDK for Java 2.x da API.

Atualizar um grupo

O exemplo de código a seguir mostra como atualizar a configuração de um grupo de Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .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);
    }
}
```

- Para obter detalhes da API, consulte [UpdateAutoScalingGroupa](#) Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com平衡amento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor Web às solicitações e verificações de integridade atualizando os parâmetros do AWS Systems Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute o cenário interativo em um prompt de comando.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\  
recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\  
server_startup_script.sh"; // Modify file location.  
    public static final String policyFile = "C:\\AWS\\resworkflow\\  
instance_policy.json"; // Modify file location.  
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\  
ssm_only_policy.json"; // Modify file location.  
    public static final String failureResponse = "doc-example-resilient-  
architecture-failure-response";  
    public static final String healthCheck = "doc-example-resilient-architecture-  
health-check";  
    public static final String templateName = "doc-example-resilience-template" ;  
    public static final String roleName = "doc-example-resilience-role";  
    public static final String policyName = "doc-example-resilience-pol";  
    public static final String profileName ="doc-example-resilience-prof" ;  
  
    public static final String badCredsProfileName ="doc-example-resilience-prof-  
bc" ;  
  
    public static final String targetGroupName = "doc-example-resilience-tg" ;  
    public static final String autoScalingGroupName = "doc-example-resilience-  
group";  
    public static final String lbName = "doc-example-resilience-lb" ;  
    public static final String protocol = "HTTP" ;  
    public static final int port = 80 ;  
  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws IOException, InterruptedException  
{  
        Scanner in = new Scanner(System.in);  
        Database database = new Database();  
        AutoScaler autoScaler = new AutoScaler();  
        LoadBalancer loadBalancer = new LoadBalancer();  
  
        System.out.println(DASHES);  
        System.out.println("Welcome to the demonstration of How to Build and Manage  
a Resilient Service!");  
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println(""""
This concludes the demo of how to build and manage a resilient service.
To keep things tidy and to avoid unwanted charges on your account, we can
clean up all AWS resources
that were created for this demo.
""");

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
Okay, we'll leave the resources intact.
Don't forget to delete them when you're done with them or you might
incur unexpected charges.
""");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}
```

```
// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database) throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName );
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(""""
        For this demo, we'll use the AWS SDK for Java (v2) to create several AWS
resources
        to set up a load-balanced web service endpoint and explore some ways to
make it resilient
        against various kinds of failures.

        Some of the resources create by this demo are:
        \t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named
"+tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
```

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

System.out.println(DASHES);
System.out.println("""
Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.

This script starts a Python web server defined in the `server.py` script.
The web server
    listens to HTTP requests on port 80 and responds to requests to '/' and to
'/healthcheck'.
    For demo purposes, this server is run as the root user. In production, the
best practice is to
        run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to assume a
role that grants
        permissions to access the DynamoDB recommendation table and Systems Manager
parameters
        that control the flow of the demo.

""");

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating an EC2 Auto Scaling group that maintains three
EC2 instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
At this point, you have EC2 instances created. Once each instance starts, it
listens for
    HTTP requests. You can see these instances in the console or continue with
the demo.

Press Enter when you're ready to continue.

""");

in.nextLine();
```

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

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load balancer. The
target group
    defines how the load balancer connects to instances. The load balancer
provides a
        single endpoint where clients connect and dispatches requests to instances
in the group.
""");

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with "+subnets.size()+" subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();
```

```
// Print the public IP address.  
System.out.println("Public IP Address: " + ipAddress);  
GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,  
ipAddress);  
if (!groupInfo.isPortOpen()) {  
    System.out.println("")  
    For this example to work, the default security group for your  
default VPC must  
        allow access from this computer. You can either add it  
automatically from this  
        example or add it yourself using the AWS Management Console.  
    "");  
  
    System.out.println("Do you want to add a rule to security group  
"+groupInfo.getGroupName() +" to allow");  
    System.out.println("inbound traffic on port "+port +" from your  
computer's IP address (y/n) ");  
    String ans = in.nextLine();  
    if ("y".equalsIgnoreCase(ans)) {  
        autoScaler.openInboundPort(groupInfo.getGroupName(),  
String.valueOf(port), ipAddress);  
        System.out.println("Security group rule added.");  
    } else {  
        System.out.println("No security group rule added.");  
    }  
}  
  
} catch (AutoScalingException e) {  
    e.printStackTrace();  
}  
} else if (wasSuccessful) {  
    System.out.println("Your load balancer is ready. You can access it by  
browsing to:");  
    System.out.println("\t http://" + elbDnsName);  
} else {  
    System.out.println("Couldn't get a successful response from the load  
balancer endpoint. Troubleshoot by");  
    System.out.println("manually verifying that your VPC and security group  
are configured correctly and that");  
    System.out.println("you can successfully make a GET request to the load  
balancer.");  
}
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }
    // A method that controls the demo part of the Java program.
    public static void demo( LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println("""
            This part of the demonstration shows how to toggle different parts of the
system
            to create situations where the web service fails, and shows how using a
resilient
            architecture can keep the web service running in spite of these failures.

            At the start, the load balancer endpoint returns recommendations and reports
that all targets are healthy.
        """);
        demoChoices(loadBalancer);

        System.out.println("""
            The web service running on the EC2 instances gets recommendations by
querying a DynamoDB table.
            The table name is contained in a Systems Manager parameter named
self.param_helper.table.
            To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.
        """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println("""
            \nNow, sending a GET request to the load balancer endpoint returns a failure
code. But, the service reports as
            healthy to the load balancer because shallow health checks don't check for
failure of the recommendation service.
        """);
        demoChoices(loadBalancer);
```

```
        System.out.println(""\");
        Instead of failing when the recommendation service fails, the web service
        can return a static response.

        While this is not a perfect solution, it presents the customer with a
        somewhat better experience than failure.

        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println(""\";
        Now, sending a GET request to the load balancer endpoint returns a static
        response.

        The service still reports as healthy because health checks are still
        shallow.

        """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println(""\";
        Let's also substitute bad credentials for one of the instances in the target
        group so that it can't
        access the DynamoDB recommendation table. We will get an instance id value.

        """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        //Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
        badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is
        "+badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is "+profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId + "
        with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
        profileAssociationId) ;

        System.out.println("")
```

```
Now, sending a GET request to the load balancer endpoint returns either a recommendation or a static response,
depending on which instance is selected by the load balancer.
""");

demoChoices(loadBalancer);

System.out.println("""
Let's implement a deep health check. For this demo, a deep health check tests whether
the web service can access the DynamoDB table that it depends on for recommendations. Note that
the deep health check is only for ELB routing and not for Auto Scaling instance health.
This kind of deep health check is not recommended for Auto Scaling instance health, because it
risks accidental termination of all instances in the Auto Scaling group when a dependent service fails.
""");

System.out.println("""
By implementing deep health checks, the load balancer can detect when one of the instances is failing
and take that instance out of rotation.
""");

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println("""
Now, checking target health indicates that the instance with bad credentials
is unhealthy. Note that it might take a minute or two for the load balancer to detect the unhealthy
instance. Sending a GET request to the load balancer endpoint always returns a recommendation, because
the load balancer takes unhealthy instances out of its rotation.
""");

demoChoices(loadBalancer);

System.out.println("""
Because the instances in this demo are controlled by an auto scaler, the simplest way to fix an unhealthy
```

```
        instance is to terminate it and let the auto scaler start a new instance to
replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
Even while the instance is terminating and the new instance is starting,
sending a GET
request to the web service continues to get a successful recommendation
response because
the load balancer routes requests to the healthy instances. After the
replacement instance
starts and reports as healthy, it is included in the load balancing
rotation.

Note that terminating and replacing an instance typically takes several
minutes, during which time you
can see the changing health check status until the new instance is running
and healthy.
""");

        demoChoices(loadBalancer);
        System.out.println("If the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    String[] actions = {
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo."
    };
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("-".repeat(88));
        System.out.println("See the current state of the service by selecting
one of the following choices:");
        for (int i = 0; i < actions.length; i++) {
            System.out.println(i + ": " + actions[i]);
        }
    }
}
```

```
    }

    try {
        System.out.print("\nWhich action would you like to take? ");
        int choice = scanner.nextInt();
        System.out.println("-".repeat(88));

        switch (choice) {
            case 0 -> {
                System.out.println("Request:\n");
                System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                CloseableHttpClient httpClient =
HttpClients.createDefault();

                // Create an HTTP GET request to the ELB.
                HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);

                // Display the JSON response
                BufferedReader reader = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder jsonResponse = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    jsonResponse.append(line);
                }
                reader.close();

                // Print the formatted JSON response.
                System.out.println("Full Response:\n");
                System.out.println(jsonResponse.toString());

                // Close the HTTP client.
                httpClient.close();
            }
            case 1 -> {
```

```
        System.out.println("\nChecking the health of load balancer
targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on
port %d is %s%n", target.target().id(), target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println(""""
Note that it can take a minute or two for the health
check to update
after changes are made.
""");
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Crie uma classe que envolva as ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {  
    private static Ec2Client ec2Client;
```

```
private static AutoScalingClient autoScalingClient;
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
```

```
/*
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
When
    * the instance is ready, Systems Manager is used to restart the Python web
server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId) throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification.builder()
        .name(newInstanceProfileName) // Make sure 'newInstanceProfileName' is a
valid IAM Instance Profile name.
        .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest.builder()
        .iamInstanceProfile(iamInstanceProfile)
        .associationId(profileAssociationId) // Make sure 'profileAssociationId'
is a valid association ID.
        .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
}
```

```
        }

        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId, newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.instanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
Collections.singletonList("cd / && sudo python3 server.py 80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
```

```
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName){
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
```

```
.instanceProfileName(profileName)
.build();

getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
System.out.println("Deleted instance profile " + profileName);

DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

// List attached role policies.
ListAttachedRolePoliciesResponse rolesResponse =
getIAMClient().listAttachedRolePolicies(role -> role.roleName(roleName));
List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
for (AttachedPolicy attachedPolicy : attachedPolicies) {
    DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(attachedPolicy.policyArn())
        .build();

    getIAMClient().detachRolePolicy(request);
    System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
}

getIAMClient().deleteRole(deleteRoleRequest);
System.out.println("Instance profile and role deleted.");

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

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name-
>name.launchTemplateName(templateName));
    System.out.format(templateName +" was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
```

```
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName +" was deleted.");
}

/*
Verify the default security group of the specified VPC allows ingress from
this
computer. This can be done by allowing ingress from this computer's IP
address. In some situations, such as connecting from a corporate network,
you
must instead specify a prefix list ID. You can also temporarily open the
port to
any IP address while running this example. If you do, be sure to remove
public
access when you're done.

*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();
}
```

```
        DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client().describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp +" is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }
            }

            if (!portIsOpen) {
                System.out.println("The inbound rule does not appear to
be open to either this computer's IP,
                    + " all IP addresses (0.0.0.0/0), or to a prefix
list ID.");
            } else {
                break;
            }
        }
    }

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
```

```
}

/*
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    The target group specifies how the load balancer forward requests to the
instances
    in the group.
*/
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest .builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to "+asGroupName);

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

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName ) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest.builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

    .map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
```

```
.collect(Collectors.toList());  
  
    String availabilityZones = String.join("", availabilityZoneNames);  
    LaunchTemplateSpecification specification =  
LaunchTemplateSpecification.builder()  
        .launchTemplateName(templateName)  
        .version("$Default")  
        .build();  
  
    String[] zones = availabilityZones.split(",");  
    CreateAutoScalingGroupRequest groupRequest =  
CreateAutoScalingGroupRequest.builder()  
        .launchTemplate(specification)  
        .availabilityZones(zones)  
        .maxSize(groupSize)  
        .minSize(groupSize)  
        .autoScalingGroupName(autoScalingGroupName)  
        .build();  
  
    try {  
        getAutoScalingClient().createAutoScalingGroup(groupRequest);  
  
    } catch (AutoScalingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Created an EC2 Auto Scaling group named "+  
autoScalingGroupName);  
    return zones;  
}  
  
public String getDefaultVPC() {  
    // Define the filter.  
    Filter defaultFilter = Filter.builder()  
        .name("is-default")  
        .values("true")  
        .build();  
  
    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =  
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest.builder()  
        .filters(defaultFilter)  
        .build();  
  
    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
```

```
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
            .map(instance -> instance.instanceId())
            .collect(Collectors.toList());
    }
}
```

```
        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
            .name("instance-id")
            .values(instanceId)
            .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest.builder()
            .filters(filter)
            .build();

        DescribeIamInstanceProfileAssociationsResponse response =
getEc2Client().describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile ) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

                software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();
                ListEntitiesForPolicyResponse listEntitiesResponse =
iamClient.listEntitiesForPolicy(listEntitiesRequest);
            }
        }
    }
}
```

```
        if (!listEntitiesResponse.policyGroups().isEmpty())
    || !listEntitiesResponse.policyUsers().isEmpty() || !
listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role
                .build();

            getIAMClient().detachRolePolicy(detachPolicyRequest);
            System.out.println("Policy detached from entities.");
        }

        // Now, you can delete the policy.
        DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                .policyArn(policy.arn())
                .build();

        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .roleName(roleName) // Remove the extra dot here
    .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
}
```

```
        System.out.println("Role " + roleName + " removed from instance profile  
" + InstanceProfile);  
    }  
  
    // Delete the instance profile after removing all roles  
    DeleteInstanceProfileRequest deleteInstanceProfileRequest =  
DeleteInstanceProfileRequest.builder()  
    .instanceProfileName(InstanceProfile)  
    .build();  
  
    getIAMClient().deleteInstanceProfile(r-  
>r.instanceProfileName(InstanceProfile));  
    System.out.println(InstanceProfile +" Deleted");  
    System.out.println("All roles and policies are deleted.");  
}  
}
```

Crie uma classe que envolva as ações do Elastic Load Balancing.

```
public class LoadBalancer {  
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;  
  
    public ElasticLoadBalancingV2Client getLoadBalancerClient() {  
        if (elasticLoadBalancingV2Client == null) {  
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
  
        return elasticLoadBalancingV2Client;  
    }  
  
    // Checks the health of the instances in the target group.  
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {  
        DescribeTargetGroupsRequest targetGroupsRequest =  
DescribeTargetGroupsRequest.builder()  
        .names(targetGroupName)  
        .build();  
  
        DescribeTargetGroupsResponse tgResponse =  
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);  
    }  
}
```

```
    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName){
    DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
    .build();

        getLoadBalancerClient().deleteLoadBalancer(builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter.waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}
```

```
// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res =
getLoadBalancerClient().describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient().deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName +" was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true ;
            } else {
                retries-- ;
                System.out.println("Got connection error from load balancer
endpoint, retrying...\"");
                TimeUnit.SECONDS.sleep(15);
            }
        }
    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }
}
```

```
        System.out.println("Status.." + success);
        return success;
    }

/*
    Creates an Elastic Load Balancing target group. The target group specifies
how
    the load balancer forward requests to instances in the group and how
instance
    health is checked.
*/
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
    Creates an Elastic Load Balancing load balancer that uses the specified
subnets
    and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port, String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
```

```
.collect(Collectors.toList());  
  
        CreateLoadBalancerRequest balancerRequest =  
CreateLoadBalancerRequest.builder()  
            .subnets(subnetIdStrings)  
            .name(lbName)  
            .scheme("internet-facing")  
            .build();  
  
        // Create and wait for the load balancer to become available.  
        CreateLoadBalancerResponse lsResponse =  
getLoadBalancerClient().createLoadBalancer(balancerRequest);  
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();  
  
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =  
getLoadBalancerClient().waiter();  
        DescribeLoadBalancersRequest request =  
DescribeLoadBalancersRequest.builder()  
            .loadBalancerArns(lbARN)  
            .build();  
  
        System.out.println("Waiting for Load Balancer " + lbName + " to become  
available.");  
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =  
loadBalancerWaiter.waitUntilLoadBalancerAvailable(request);  
        waiterResponse.matched().response().ifPresent(System.out::println);  
        System.out.println("Load Balancer " + lbName + " is available.");  
  
        // Get the DNS name (endpoint) of the load balancer.  
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();  
        System.out.println("**** Load Balancer DNS Name: " + lbDNSName);  
  
        // Create a listener for the load balance.  
        Action action = Action.builder()  
            .targetGroupArn(targetGroupARN)  
            .type("forward")  
            .build();  
  
        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()  
.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())  
            .defaultActions(action)  
            .port(port)  
            .protocol(protocol)
```

```
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println( "Created listener to forward traffic from load
balancer " + lbName + " to target group " + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName){
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

            getDynamoDbClient().describeTable(describeTableRequest);
        }
    }
}
```

```
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/*
Creates a DynamoDB table to use a recommendation service. The table has a
hash key named 'MediaType' that defines the type of media recommended, such
as
    Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
    forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
    }
}
```

```
.provisionedThroughput()
    ProvisionedThroughput.builder()
        .readCapacityUnits(5L)
        .writeCapacityUnits(5L)
        .build())
    .build();

getDynamoDbClient().createTable(createTableRequest);
System.out.println("Creating table " + tableName + "...");

// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}

}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table->table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
```

```
String mediaType = currentNode.path("MediaType").path("S").asText();
int itemId = currentNode.path("ItemId").path("N").asInt();
String title = currentNode.path("Title").path("S").asText();
String creator = currentNode.path("Creator").path("S").asText();

// Create a Recommendation object and set its properties.
Recommendation rec = new Recommendation();
rec.setMediaType(mediaType);
rec.setItemId(itemId);
rec.setTitle(title);
rec.setCreator(creator);

// Put the item into the DynamoDB table.
mappedTable.putItem(rec); // Add the Recommendation to the list.
}
System.out.println("Added all records to the "+tableName);
}
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName );
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
    }
}
```

```
        .overwrite(true)
        .type("String")
        .build();

    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeElbInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)

- [ReplaceElbInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Gerencie grupos e instâncias

O código de exemplo a seguir mostra como:

- Crie um grupo do Amazon EC2 Auto Scaling com um modelo de lançamento e zonas de disponibilidade e obtenha informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualize a capacidade desejada do grupo e aguarde a inicialização de uma instância.
- Encerre uma instância no grupo.
- Liste as atividades de escalabilidade que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas de CloudWatch métricas e, em seguida, limpe os recursos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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> <vpcZoneId>\n\n" +
            "Where:\n" +
            "      groupName - The name of the Auto Scaling group.\n" +
            "      launchTemplateName - The name of the launch template. \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 != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        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,
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);
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 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)
                .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) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .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);
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)

- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

CloudFront exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudFront.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar uma distribuição

O exemplo de código a seguir mostra como criar uma CloudFront distribuição.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

O exemplo a seguir usa um bucket do Amazon Simple Storage Service (Amazon S3) como origem de conteúdo.

Depois de criar a distribuição, o código cria um [CloudFrontWaiter](#) para esperar até que a distribuição seja implantada antes de retornar a distribuição.

```
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;
}
}
```

- Para obter detalhes da API, consulte [CreateDistribution](#) Referência AWS SDK for Java 2.x da API.

Criar uma função

O exemplo de código a seguir mostra como criar uma CloudFront função da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateFunction](#) Referência AWS SDK for Java 2.x da API.

Crie um grupo-chave

O exemplo de código a seguir mostra como criar um grupo de chaves que você pode usar com URLs assinados e cookies assinados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Um grupo de chaves exige pelo menos uma chave pública usada para verificar URLs ou cookies assinados.

```
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;
    }
}
```

- Para obter detalhes da API, consulte [CreateKeyGroup](#) Referência AWS SDK for Java 2.x da API.

Excluir uma distribuição

O exemplo de código a seguir mostra como excluir uma CloudFront distribuição.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

O exemplo de código a seguir atualiza uma distribuição para desativada, usa um garçom que aguarda a implantação da alteração e, em seguida, exclui a distribuição.

```
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);
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [DeleteDistribution](#)
 - [UpdateDistribution](#)

Excluir recursos de assinatura

O exemplo de código a seguir mostra como excluir recursos que são usados para obter acesso a conteúdo restrito em um bucket do Amazon Simple Storage Service (Amazon S3).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.

- [DeleteKeyGroup](#)
- [DeleteOriginAccessControl](#)
- [DeletePublicKey](#)

Atualizar uma distribuição

O exemplo de código a seguir mostra como atualizar uma CloudFront distribuição da Amazon.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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())  
    } catch (CloudFrontException e) {  
        System.out.println(e.getMessage());  
    }  
}
```

```
        .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);
    }
}
```

- Para obter detalhes da API, consulte [UpdateDistribution](#) na Referência AWS SDK for Java 2.x da API.

Carregar uma chave pública

O exemplo de código a seguir mostra como fazer upload de uma chave pública.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

O exemplo de código a seguir lê uma chave pública e a carrega na 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);
        }
    }
}
```

- Para obter detalhes da API, consulte [CreatePublicKey](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Assine URLs e cookies

O exemplo de código a seguir mostra como criar URLs e cookies assinados que permitem acesso a recursos restritos.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Use a [CannedSignerRequest](#) classe para assinar URLs ou cookies com uma política predefinida.

```
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 a [CustomSignerRequest](#) classe para assinar URLs ou cookies com uma política personalizada. Os `activeDate` e `ipRange` são métodos opcionais.

```
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();
    }
}
```

```
}
```

O exemplo a seguir demonstra o uso da [CloudFrontUtilities](#) classe para produzir cookies e URLs assinados. [Veja](#) este exemplo de código em 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;  
}  
}
```

- Para obter detalhes da API, consulte [CloudFrontUtilities](#) Referência AWS SDK for Java 2.x da API.

CloudWatch exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá CloudWatch

O exemplo de código a seguir mostra como começar a usar o CloudWatch.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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);
    }
}
}
```

- Para obter detalhes da API, consulte [ListMetrics](#) a Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um painel

O exemplo de código a seguir mostra como criar um CloudWatch painel da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [PutDashboard](#) a Referência AWS SDK for Java 2.x da API.

Crie um alarme de métrica

O exemplo de código a seguir mostra como criar ou atualizar um CloudWatch alarme da Amazon e associá-lo à métrica especificada, à expressão matemática métrica, ao modelo de detecção de anomalias ou à consulta do Metrics Insights.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [PutMetricAlarma](#) Referência AWS SDK for Java 2.x da API.

Criar um detector de anomalias

O exemplo de código a seguir mostra como criar um detector de CloudWatch anomalias da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [PutAnomalyDetector](#) a Referência AWS SDK for Java 2.x da API.

Excluir alarmes

O exemplo de código a seguir mostra como excluir CloudWatch alarmes da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteCWAAlarm(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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteAlarms](#) Referência AWS SDK for Java 2.x da API.

Excluir um detector de anomalias

O exemplo de código a seguir mostra como excluir um detector de CloudWatch anomalias da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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();
}
}
```

- Para obter detalhes da API, consulte [DeleteAnomalyDetector](#) Referência AWS SDK for Java 2.x da API.

Excluir painéis

O exemplo de código a seguir mostra como excluir CloudWatch painéis da Amazon.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteDashboards](#) a Referência AWS SDK for Java 2.x da API.

Descrever o histórico do alarme

O exemplo de código a seguir mostra como descrever um histórico de CloudWatch alarms da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DescribeAlarmHistory](#) Referência AWS SDK for Java 2.x da API.

Descrever alarmes

O exemplo de código a seguir mostra como descrever os CloudWatch alarmes da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeAlarms](#) a Referência AWS SDK for Java 2.x da API.

Descrever alarmes para uma métrica

O exemplo de código a seguir mostra como descrever os CloudWatch alarmes da Amazon para uma métrica.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeAlarmsForMetrica](#) Referência AWS SDK for Java 2.x da API.

Descrever detectores de anomalias

O exemplo de código a seguir mostra como descrever os detectores de CloudWatch anomalias da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DescribeAnomalyDetectors](#) Referência AWS SDK for Java 2.x da API.

Desabilitar ações de alarme

O exemplo de código a seguir mostra como desativar as ações de CloudWatch alarme da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DisableAlarmActions](#) a Referência AWS SDK for Java 2.x da API.

Habilitar ações de alarme

O exemplo de código a seguir mostra como ativar as ações de CloudWatch alarme da Amazon.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [EnableAlarmActions](#) Referência AWS SDK for Java 2.x da API.

Obter uma imagem de dados de métricas

O exemplo de código a seguir mostra como obter uma imagem de dados CloudWatch métricos da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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\",  
            \"view\": \"timeSeries\",  
            \"stacked\": false,  
            \"period\": 10,  
            \"width\": 1400,  
            \"height\": 600,  
            \"metrics\": [\n                \"AWS/Billing\",  
                \"EstimatedCharges\",  
                \"Currency\",  
                \"USD\"\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);
    }
}
```

- Para obter detalhes da API, consulte [GetMetricWidgetImage](#) Referência AWS SDK for Java 2.x da API.

Obter dados de métricas

O exemplo de código a seguir mostra como obter dados CloudWatch métricos da Amazon.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
}  
}
```

- Para obter detalhes da API, consulte [GetMetricData](#) a Referência AWS SDK for Java 2.x da API.

Obter estatísticas de métricas

O exemplo de código a seguir mostra como obter estatísticas CloudWatch métricas da Amazon.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [GetMetricStatistics](#) a Referência AWS SDK for Java 2.x da API.

Listar painéis

O exemplo de código a seguir mostra como listar os CloudWatch painéis da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [ListDashboards](#) a Referência AWS SDK for Java 2.x da API.

Listar as métricas

O exemplo de código a seguir mostra como listar os metadados das CloudWatch métricas da Amazon. Para obter dados de uma métrica, use as GetMetricStatistics ações GetMetricData ou.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [ListMetrics](#) a Referência AWS SDK for Java 2.x da API.

Inserir dados em uma métrica

O exemplo de código a seguir mostra como publicar pontos de dados métricos na Amazon CloudWatch.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 metric " +
+customMetricName);

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

- Para obter detalhes da API, consulte [PutMetricData](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Começar a usar métricas, painéis e alarmes

O código de exemplo a seguir mostra como:

- Listar CloudWatch namespaces e métricas.
- Obtenha estatísticas para uma métrica e para faturamento estimado.
- Crie e atualize um painel.
- Crie e adicione dados a uma métrica.
- Crie e acione um alarme e, em seguida, visualize o histórico de alarmes.
- Crie um detector de anomalias.
- Obtenha uma imagem de métrica e, em seguida, limpe os recursos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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);
deleteCWAAlarm(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 deleteCWAAlarm(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                "
                + " \"region\": \"us-east-1\",\\n                "
                + " \"start\": \"2018-01-01T00:00:00Z\",\\n                "
                + " \"end\": \"2018-01-01T00:00:00Z\",\\n                "
                + " \"metricName\": \"CPUUtilization\",\\n                "
                + " \"dimensions\": {\\n                    \"MetricName\": \"CPUUtilization\",\\n                    \"Dimensions\": [\\n                        {\\n                            \"Name\": \"Region\",\\n                            \"Value\": \"us-east-1\"\\n                        },\\n                        {\\n                            \"Name\": \"InstanceId\",\\n                            \"Value\": \"i-000000000000000000\"\\n                        }\\n                    ]\\n                }\\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" +
    "};\n\n    GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
    .metricWidget(myJSON)
    .build();\n\n    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);
    }
\n} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}\n\npublic 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 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;  
}  
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [DeleteAlarms](#)
 - [DeleteAnomalyDetector](#)
 - [DeleteDashboards](#)
 - [DescribeAlarmHistory](#)
 - [DescribeAlarms](#)
 - [DescribeAlarmsForMetric](#)
 - [DescribeAnomalyDetectors](#)
 - [GetMetricData](#)
 - [GetMetricStatistics](#)
 - [GetMetricWidgetImage](#)

- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

CloudWatch Exemplos de eventos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudWatch Events.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Adição de um destino

O exemplo de código a seguir mostra como adicionar um alvo a um evento da Amazon CloudWatch Events.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void putCWTTargets(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);
    }
}
```

- Para obter detalhes da API, consulte [PutTargets](#) a Referência AWS SDK for Java 2.x da API.

Criar uma regra agendada

O exemplo de código a seguir mostra como criar uma regra programada de CloudWatch eventos da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [PutRule](#) a Referência AWS SDK for Java 2.x da API.

Enviar eventos

O exemplo de código a seguir mostra como enviar CloudWatch eventos da Amazon Events.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [PutEvents](#) Referência AWS SDK for Java 2.x da API.

CloudWatch Exemplos de registros usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudWatch Logs.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Criar um filtro de assinatura

O exemplo de código a seguir mostra como criar um filtro de assinatura do Amazon CloudWatch Logs.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [PutSubscriptionFilter](#) Referência AWS SDK for Java 2.x da API.

Excluir um filtro de assinatura

O exemplo de código a seguir mostra como excluir um filtro de assinatura do Amazon CloudWatch Logs.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteSubscriptionFilter](#) Referência AWS SDK for Java 2.x da API.

Descreva os filtros de assinatura existentes

O exemplo de código a seguir mostra como descrever os filtros de assinatura existentes do Amazon CloudWatch Logs.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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.println(filter);  
            }  
        }  
    } catch (AmazonServiceException ase) {  
        System.out.println("Caught an AmazonServiceException, which " +  
            "means your request was well-formed but did not " +  
            "succeed due to semantic errors or other transient " +  
            "issues. Status code: " + ase.getStatusCode());  
        System.out.println("Error Message: " + ase.getErrorMessage());  
        System.out.println("HTTP Status Code: " + ase.getStatusCode());  
        System.out.println("AWS Request ID: " + ase.getRequestId());  
    } catch (AmazonClientException ace) {  
        System.out.println("Caught an AmazonClientException, which " +  
            "means the client encountered an internal error while " +  
            "communicating with Amazon. This exception " +  
            "typically indicates that you've made a call " +  
            "that failed on the server side.  
    }  
}
```

```
        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");  
}
```

- Para obter detalhes da API, consulte [DescribeSubscriptionFilters](#) Referência AWS SDK for Java 2.x da API.

Exemplos de provedores de identidade do Amazon Cognito usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Confirmar um usuário

O exemplo de código a seguir mostra como confirmar um usuário do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ConfirmSignUp](#) Referência AWS SDK for Java 2.x da API.

Obter um token para associar uma aplicação de MFA a um usuário

O exemplo de código a seguir mostra como obter um token para associar um aplicativo de MFA a um usuário do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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();  
}
```

- Para obter detalhes da API, consulte [AssociateSoftwareToken](#) a Referência AWS SDK for Java 2.x da API.

Ter informações sobre um usuário

O exemplo de código a seguir mostra como obter informações sobre um usuário do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void getAdminUser(CognitoIdentityProviderClient  
identityProviderClient, String userName, String poolId) {  
    try {  
        Admin GetUserRequest userRequest = Admin GetUserRequest.builder()  
            .username(userName)  
            .userPoolId(poolId)  
            .build();  
  
        Admin GetUserResponse response =  
        identityProviderClient.admin GetUser(userRequest);  
        System.out.println("User status "+response.userStatusAsString());  
  
    } catch (CognitoIdentityProviderException e){  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [Admin GetUser](#) Referência AWS SDK for Java 2.x da API.

Listar usuários

O exemplo de código a seguir mostra como listar usuários do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [ListUsers](#) a Referência AWS SDK for Java 2.x da API.

Reenviar um código de confirmação

O exemplo de código a seguir mostra como reenviar um código de confirmação do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ResendConfirmationCode](#) a Referência AWS SDK for Java 2.x da API.

Responder a um desafio de autenticação

O exemplo de código a seguir mostra como responder a um desafio de autenticação do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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());  
}
```

- Para obter detalhes da API, consulte [AdminRespondToAuthChallenge](#) a Referência AWS SDK for Java 2.x da API.

Inscrever um usuário

O exemplo de código a seguir mostra como cadastrar um usuário no Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [SignUp](#) Referência AWS SDK for Java 2.x da API.

Iniciar a autenticação com credenciais de administrador

O exemplo de código a seguir mostra como iniciar a autenticação com o Amazon Cognito e as credenciais de administrador.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

- Para obter detalhes da API, consulte [AdminInitiateAuth](#) Referência AWS SDK for Java 2.x da API.

Verificar uma aplicação de MFA com um usuário

O exemplo de código a seguir mostra como verificar um aplicativo de MFA com um usuário do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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);  
    }  
}
```

- Para obter detalhes da API, consulte [VerifySoftwareToken](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O código de exemplo a seguir mostra como:

- Inscreva e confirme um usuário com nome de usuário, senha e endereço de e-mail.
- Configure a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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 {
        Admin GetUserRequest userRequest = Admin GetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        Admin GetUserResponse response =
identityProviderClient.admin GetUser(userRequest);
        System.out.println("User status "+response.userStatusAsString());

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [Admin GetUser](#)
 - [Admin Initiate Auth](#)
 - [Admin Respond To Auth Challenge](#)
 - [Associate Software Token](#)
 - [Confirm Device](#)
 - [Confirm Sign Up](#)
 - [Initiate Auth](#)
 - [List Users](#)
 - [Resend Confirmation Code](#)
 - [Respond To Auth Challenge](#)
 - [Sign Up](#)
 - [Verify Software Token](#)

Exemplos do Amazon Comprehend usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Comprehend.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Criar um classificador de documentos

O exemplo de código a seguir mostra como criar um classificador de documentos Amazon Comprehend.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateDocumentClassifier](#) Referência AWS SDK for Java 2.x da API.

Detectar entidades em um documento

O exemplo de código a seguir mostra como detectar entidades em um documento com o Amazon Comprehend.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [DetectEntities](#) Referência AWS SDK for Java 2.x da API.

Detecte frases-chave em um documento

O exemplo de código a seguir mostra como detectar frases-chave em um documento com o Amazon Comprehend.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

```
}
```

- Para obter detalhes da API, consulte [DetectKeyPhrases](#)a Referência AWS SDK for Java 2.x da API.

Detecte elementos sintáticos de um documento

O exemplo de código a seguir mostra como detectar elementos sintáticos de um documento com o Amazon Comprehend.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

```
}
```

- Para obter detalhes da API, consulte [DetectSyntaxa Referência AWS SDK for Java 2.x da API](#).

Detecte o idioma dominante em um documento

O exemplo de código a seguir mostra como detectar a linguagem dominante em um documento com o Amazon Comprehend.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DetectDominantLanguage](#) Referência AWS SDK for Java 2.x da API.

Detecte o sentimento de um documento

O exemplo de código a seguir mostra como detectar o sentimento de um documento com o Amazon Comprehend.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DetectSentiment](#) Referência AWS SDK for Java 2.x da API.

Exemplos do DynamoDB usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o DynamoDB.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar uma tabela

O exemplo de código a seguir mostra como criar uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateTablea Referência AWS SDK for Java 2.x da API](#).

Excluir uma tabela

O exemplo de código a seguir mostra como excluir uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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!");  
}
```

- Para obter detalhes da API, consulte [DeleteTable](#) a Referência AWS SDK for Java 2.x da API.

Excluir um item de uma tabela

O exemplo de código a seguir mostra como excluir um item de uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteItem](#) na Referência AWS SDK for Java 2.x da API.

Obter um item de uma tabela

O exemplo de código a seguir mostra como obter um item de uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Obtém um item de uma tabela usando DynamoDbClient o.

```
public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal ) {
    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
```

```
.s(keyVal)
.build();

GetItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName(tableName)
    .build();

try {
    // If there is no matching item, GetItem does not return any data.
    Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();
    if (returnedItem.isEmpty())
        System.out.format("No item found with the key %s!\n", key);
    else {
        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());
        }
    }
}

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

- Para obter detalhes da API, consulte [GetItem](#) Referência AWS SDK for Java 2.x da API.

Obter informações sobre uma tabela

O exemplo de código a seguir mostra como obter informações sobre uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void describeDynamoDBTable(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!");  
}
```

- Para obter detalhes da API, consulte [DescribeTable](#) Referência AWS SDK for Java 2.x da API.

Listar tabelas

O exemplo de código a seguir mostra como listar tabelas do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
            }  
        } catch (AmazonServiceException ase) {  
            System.out.println("Caught an AmazonServiceException, which " +  
                "means your request was well-formed but did not succeed  
                due to semantic errors." + ase.getMessage());  
        } catch (AmazonClientException ace) {  
            System.out.println("Caught an AmazonClientException, which " +  
                "means the client could not reach Amazon. " +  
                "This might be due to low network connection errors," +  
                "� or a configuration problem." + ace.getMessage());  
        }  
    }  
}
```

```
        lastName = response.lastEvaluatedTableName();
        if (lastName == null) {
            moreTables = false;
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- Para obter detalhes da API, consulte [ListTables](#) Referência AWS SDK for Java 2.x da API.

Colocar um item em uma tabela

O exemplo de código a seguir mostra como colocar um item em uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Coloca um item em uma tabela usando [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);
    }
}
```

- Para obter detalhes da API, consulte [PutItem](#) Referência AWS SDK for Java 2.x da API.

Consultar uma tabela

O exemplo de código a seguir mostra como consultar uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Consulta uma tabela usando [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;
}
```

Consulta uma tabela usando o [DynamoDbClient](#) e um índice secundário.

```
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);
    }
}
```

- Para obter detalhes da API, consulte [Query](#) na Referência da API do AWS SDK for Java 2.x.

Verificar uma tabela

O exemplo de código a seguir mostra como escanear uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Escaneia uma tabela do Amazon DynamoDB [DynamoDbClient](#) usando.

```
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);
}
}
```

- Para obter detalhes da API, consulte [Scan](#) na Referência da API do AWS SDK for Java 2.x.

Atualizar um item em uma tabela

O exemplo de código a seguir mostra como atualizar um item em uma tabela do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Atualiza um item em uma tabela usando [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!");
}
```

- Para obter detalhes da API, consulte [UpdateItem](#) Referência AWS SDK for Java 2.x da API.

Gravar um lote de itens

O exemplo de código a seguir mostra como gravar um lote de itens do DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Insira vários itens em uma tabela usando o cliente aprimorado.

```
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(  
                ...  
            );  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    }  
}
```

```
        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);
    }
}
```

- Para obter detalhes da API, consulte [BatchWriteItem](#) Referência AWS SDK for Java 2.x da API.

Cenários

Conceitos básicos de tabelas, itens e consultas

O código de exemplo a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.

- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Exclua um filme da tabela e, depois, exclua a tabela.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie uma tabela do DynamoDB.

```
// 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);  
}  
}
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
// 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++;
}
}
```

Obtenha um item de uma tabela.

```
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);
}
}
```

Exemplo completo.

```
/**
 * 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);
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)

- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

Consultar uma tabela usando lotes de instruções PartiQL

O código de exemplo a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
```

- Para obter detalhes da API, consulte [BatchExecuteStatement](#) Referência AWS SDK for Java 2.x da API.

Consultar uma tabela usando o PartiQL

O código de exemplo a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public class ScenarioPartiQ {

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

        final String usage = "\n" +
            "Usage:\n" +
            "      <fileName>\n\n" +
```

```
"Where:\n" +\n        "    fileName - The path to the moviedata.json file that you can\ndownload from the Amazon DynamoDB Developer Guide.\n" ;\n\n    if (args.length != 1) {\n        System.out.println(usage);\n        System.exit(1);\n    }\n\n    String fileName = args[0];\n    String tableName = "MoviesPartiQ";\n    ProfileCredentialsProvider credentialsProvider =\nProfileCredentialsProvider.create();\n    Region region = Region.US_EAST_1;\n    DynamoDbClient ddb = DynamoDbClient.builder()\n        .credentialsProvider(credentialsProvider)\n        .region(region)\n        .build();\n\n    System.out.println("***** Creating an Amazon DynamoDB table named\nMoviesPartiQ with a key named year and a sort key named title.");\n    createTable(ddb, tableName);\n\n    System.out.println("***** Loading data into the MoviesPartiQ table.");\n    loadData(ddb, fileName);\n\n    System.out.println("***** Getting data from the MoviesPartiQ table.");\n    getItem(ddb);\n\n    System.out.println("***** Putting a record into the MoviesPartiQ table.");\n    putRecord(ddb);\n\n    System.out.println("***** Updating a record.");\n    updateTableItem(ddb);\n\n    System.out.println("***** Querying the movies released in 2013.");\n    queryTable(ddb);\n\n    System.out.println("***** Deleting the Amazon DynamoDB table.");\n    deleteDynamoDBTable(ddb, tableName);\n    ddb.close();\n}\n\npublic 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());
}
}
```

- Para obter detalhes da API, consulte [ExecuteStatement](#) na Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon EC2 usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon EC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Hello Amazon EC2

Os exemplos de código a seguir mostram como começar a usar o Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
        }
    }
```

- Para obter detalhes da API, consulte [DescribeSecurityGroups](#) Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Alocar um endereço IP elástico

O exemplo de código a seguir mostra como alocar um endereço IP elástico para o Amazon EC2.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [AllocateAddress](#) Referência AWS SDK for Java 2.x da API.

Para associar um endereço IP elástico a uma instância

O exemplo de código a seguir mostra como associar um endereço IP elástico a uma instância do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [AssociateAddress](#) a Referência AWS SDK for Java 2.x da API.

Crie um grupo de segurança

O exemplo de código a seguir mostra como criar um grupo de segurança do Amazon EC2.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateSecurityGroup](#) na Referência AWS SDK for Java 2.x da API.

Criar um par de chaves de segurança

O exemplo de código a seguir mostra como criar um par de chaves de segurança para o Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [CreateKeyPair](#) Referência AWS SDK for Java 2.x da API.

Criar e executar uma instância

O exemplo de código a seguir mostra como criar e executar uma instância do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

```
}
```

- Para obter detalhes da API, consulte [RunInstances](#) Referência AWS SDK for Java 2.x da API.

Excluir um grupo de segurança

O exemplo de código a seguir mostra como excluir um grupo de segurança do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteSecurityGroup](#) Referência AWS SDK for Java 2.x da API.

Excluir um par de chaves de segurança

O exemplo de código a seguir mostra como excluir um par de chaves de segurança do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteKeyPair](#) Referência AWS SDK for Java 2.x da API.

Descrever instâncias

O exemplo de código a seguir mostra como descrever as instâncias do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [DescribeInstances](#) Referência AWS SDK for Java 2.x da API.

Desassociar um endereço IP elástico de uma instância

O exemplo de código a seguir mostra como desassociar um endereço IP elástico de uma instância do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DisassociateAddress](#) Referência AWS SDK for Java 2.x da API.

Obter dados sobre um grupo de segurança

O exemplo de código a seguir mostra como obter dados sobre um grupo de segurança do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeSecurityGroups](#) a Referência AWS SDK for Java 2.x da API.

Obter dados sobre tipos de instâncias

O exemplo de código a seguir mostra como obter dados sobre os tipos de instância do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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 "";
    }
```

- Para obter detalhes da API, consulte [DescribeInstanceTypes](#) Referência AWS SDK for Java 2.x da API.

Listar pares de chaves de segurança

O exemplo de código a seguir mostra como listar os pares de chaves de segurança do Amazon EC2. SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeKeyPairs](#) a Referência AWS SDK for Java 2.x da API.

Liberar um endereço IP elástico

O exemplo de código a seguir mostra como liberar um endereço IP elástico.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ReleaseAddress](#) Referência AWS SDK for Java 2.x da API.

Definir regras de entrada para um grupo de segurança

O exemplo de código a seguir mostra como definir regras de entrada para um grupo de segurança do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [AuthorizeSecurityGroupIngress](#) Referência AWS SDK for Java 2.x da API.

Iniciar uma instância

O exemplo de código a seguir mostra como iniciar uma instância do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }
```

- Para obter detalhes da API, consulte [StartInstances](#) Referência AWS SDK for Java 2.x da API.

Interromper uma instância

O exemplo de código a seguir mostra como parar uma instância do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
```

- Para obter detalhes da API, consulte [StopInstances](#) a Referência AWS SDK for Java 2.x da API.

Como encerrar uma instância

O exemplo de código a seguir mostra como encerrar uma instância do Amazon EC2.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [TerminateInstances](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com平衡amento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor Web às solicitações e verificações de integridade atualizando os parâmetros do AWS Systems Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-health-check";
    public static final String templateName = "doc-example-resilience-template" ;
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName ="doc-example-resilience-prof" ;

    public static final String badCredsProfileName ="doc-example-resilience-profile" ;

    public static final String targetGroupName = "doc-example-resilience-tg" ;
    public static final String autoScalingGroupName = "doc-example-resilience-group";
    public static final String lbName = "doc-example-resilience-lb" ;
    public static final String protocol = "HTTP" ;
    public static final int port = 80 ;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws IOException, InterruptedException
    {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage a Resilient Service!");
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println(""""
This concludes the demo of how to build and manage a resilient service.
To keep things tidy and to avoid unwanted charges on your account, we can
clean up all AWS resources
that were created for this demo.
""");

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
    Okay, we'll leave the resources intact.
    Don't forget to delete them when you're done with them or you might
incur unexpected charges.
""");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
```

```
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database) throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("**** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName );
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println("""
        For this demo, we'll use the AWS SDK for Java (v2) to create several AWS
resources
        to set up a load-balanced web service endpoint and explore some ways to
make it resilient
        against various kinds of failures.

        Some of the resources create by this demo are:
        \t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named
"+tableName);
    Database database = new Database();
```

```
database.createTable(tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.

    This script starts a Python web server defined in the `server.py` script.
The web server
    listens to HTTP requests on port 80 and responds to requests to '/' and to
'/healthcheck'.

    For demo purposes, this server is run as the root user. In production, the
best practice is to
        run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to assume a
role that grants
        permissions to access the DynamoDB recommendation table and Systems Manager
parameters
        that control the flow of the demo.

""");

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating an EC2 Auto Scaling group that maintains three
EC2 instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
At this point, you have EC2 instances created. Once each instance starts, it
listens for
    HTTP requests. You can see these instances in the console or continue with
the demo.

Press Enter when you're ready to continue.

""");
```

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

    System.out.println(DASHES);
    System.out.println("Creating variables that control the flow of the demo.");
    ParameterHelper paramHelper = new ParameterHelper();
    paramHelper.reset();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an Elastic Load Balancing target group and load balancer. The
target group
        defines how the load balancer connects to instances. The load balancer
provides a
        single endpoint where clients connect and dispatches requests to instances
in the group.
        """);

    String vpcId = autoScaler.getDefaultVPC();
    List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
    System.out.println("You have retrieved a list with "+subnets.size() +" "
subnets);
    String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
    String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
    autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
    System.out.println("Verifying access to the load balancer endpoint...");
    boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessful) {
        System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to "http://checkip.amazonaws.com"
        HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
        try {
            // Execute the request and get the response
            HttpResponse response = httpClient.execute(httpGet);

            // Read the response content.
```

```
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
For this example to work, the default security group for your
default VPC must
allow access from this computer. You can either add it
automatically from this
example or add it yourself using the AWS Management Console.
""");

            System.out.println("Do you want to add a rule to security group
"+groupInfo.getGroupName() +" to allow");
            System.out.println("inbound traffic on port "+port +" from your
computer's IP address (y/n) ");
            String ans = in.nextLine();
            if ("y".equalsIgnoreCase(ans)) {
                autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                System.out.println("Security group rule added.");
            } else {
                System.out.println("No security group rule added.");
            }
        }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}
```

```
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}
// A method that controls the demo part of the Java program.
public static void demo( LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println("""
        This part of the demonstration shows how to toggle different parts of the
system
        to create situations where the web service fails, and shows how using a
resilient
        architecture can keep the web service running in spite of these failures.

        At the start, the load balancer endpoint returns recommendations and reports
that all targets are healthy.
""");
    demoChoices(loadBalancer);

    System.out.println("""
        The web service running on the EC2 instances gets recommendations by
querying a DynamoDB table.
        The table name is contained in a Systems Manager parameter named
self.param_helper.table.
        To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.
""");
    paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

    System.out.println("""
        \nNow, sending a GET request to the load balancer endpoint returns a failure
code. But, the service reports as
        healthy to the load balancer because shallow health checks don't check for
failure of the recommendation service.
""");
```

```
        demoChoices(loadBalancer);

        System.out.println("""
            Instead of failing when the recommendation service fails, the web service
            can return a static response.
            While this is not a perfect solution, it presents the customer with a
            somewhat better experience than failure.
        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
            Now, sending a GET request to the load balancer endpoint returns a static
            response.
            The service still reports as healthy because health checks are still
            shallow.
        """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
            Let's also substitute bad credentials for one of the instances in the target
            group so that it can't
            access the DynamoDB recommendation table. We will get an instance id value.
        """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        //Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is
"+badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is "+profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId + " "
with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId) ;
```

```
System.out.println(""\");
Now, sending a GET request to the load balancer endpoint returns either a
recommendation or a static response,
depending on which instance is selected by the load balancer.
""");

demoChoices(loadBalancer);

System.out.println(""\";
Let's implement a deep health check. For this demo, a deep health check
tests whether
the web service can access the DynamoDB table that it depends on for
recommendations. Note that
the deep health check is only for ELB routing and not for Auto Scaling
instance health.
This kind of deep health check is not recommended for Auto Scaling instance
health, because it
risks accidental termination of all instances in the Auto Scaling group when
a dependent service fails.
""");

System.out.println(""\";
By implementing deep health checks, the load balancer can detect when one of
the instances is failing
and take that instance out of rotation.
""");

paramHelper.put(paramHelper.healthCheck, "deep");

System.out.println(""\";
Now, checking target health indicates that the instance with bad
credentials
is unhealthy. Note that it might take a minute or two for the load balancer
to detect the unhealthy
instance. Sending a GET request to the load balancer endpoint always returns
a recommendation, because
the load balancer takes unhealthy instances out of its rotation.
""");

demoChoices(loadBalancer);

System.out.println(""\";
Because the instances in this demo are controlled by an auto scaler, the
simplest way to fix an unhealthy
```

```
        instance is to terminate it and let the auto scaler start a new instance to
replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
Even while the instance is terminating and the new instance is starting,
sending a GET
request to the web service continues to get a successful recommendation
response because
the load balancer routes requests to the healthy instances. After the
replacement instance
starts and reports as healthy, it is included in the load balancing
rotation.

Note that terminating and replacing an instance typically takes several
minutes, during which time you
can see the changing health check status until the new instance is running
and healthy.
""");

        demoChoices(loadBalancer);
        System.out.println("If the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    String[] actions = {
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo."
    };
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("-".repeat(88));
        System.out.println("See the current state of the service by selecting
one of the following choices:");
        for (int i = 0; i < actions.length; i++) {
            System.out.println(i + ": " + actions[i]);
        }
    }
}
```

```
}

try {
    System.out.print("\nWhich action would you like to take? ");
    int choice = scanner.nextInt();
    System.out.println("-".repeat(88));

    switch (choice) {
        case 0 -> {
            System.out.println("Request:\n");
            System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
            CloseableHttpClient httpClient =
HttpClients.createDefault();

            // Create an HTTP GET request to the ELB.
            HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);

            // Display the JSON response
            BufferedReader reader = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
            StringBuilder jsonResponse = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                jsonResponse.append(line);
            }
            reader.close();

            // Print the formatted JSON response.
            System.out.println("Full Response:\n");
            System.out.println(jsonResponse.toString());

            // Close the HTTP client.
            httpClient.close();
        }
        case 1 -> {
```

```

        System.out.println("\nChecking the health of load balancer
targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on
port %d is %s%n", target.target().id(), target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
Note that it can take a minute or two for the health
check to update
after changes are made.
""");
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
}

```

Crie uma classe que envolva as ações do Auto Scaling e do Amazon EC2.

```

public class AutoScaler {

    private static Ec2Client ec2Client;
}

```

```
private static AutoScalingClient autoScalingClient;
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
```

```
/*
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/***
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
When
    * the instance is ready, Systems Manager is used to restart the Python web
server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId) throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification.builder()
        .name(newInstanceProfileName) // Make sure 'newInstanceProfileName' is a
valid IAM Instance Profile name.
        .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest.builder()
        .iamInstanceProfile(iamInstanceProfile)
        .associationId(profileAssociationId) // Make sure 'profileAssociationId'
is a valid association ID.
        .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
}
```

```
        }

        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId, newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.instanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
Collections.singletonList("cd / && sudo python3 server.py 80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
```

```
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName){
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
```

```
.instanceProfileName(profileName)
.build();

getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
System.out.println("Deleted instance profile " + profileName);

DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

// List attached role policies.
ListAttachedRolePoliciesResponse rolesResponse =
getIAMClient().listAttachedRolePolicies(role -> role.roleName(roleName));
List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
for (AttachedPolicy attachedPolicy : attachedPolicies) {
    DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(attachedPolicy.policyArn())
        .build();

    getIAMClient().detachRolePolicy(request);
    System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
}

getIAMClient().deleteRole(deleteRoleRequest);
System.out.println("Instance profile and role deleted.");

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

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name-
>name.launchTemplateName(templateName));
    System.out.format(templateName +" was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
```

```
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName +" was deleted.");
}

/*
Verify the default security group of the specified VPC allows ingress from
this
computer. This can be done by allowing ingress from this computer's IP
address. In some situations, such as connecting from a corporate network,
you
must instead specify a prefix list ID. You can also temporarily open the
port to
any IP address while running this example. If you do, be sure to remove
public
access when you're done.

*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();
}
```

```
        DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client().describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp +" is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }
            }

            if (!portIsOpen) {
                System.out.println("The inbound rule does not appear to
be open to either this computer's IP,
                    + " all IP addresses (0.0.0.0/0), or to a prefix
list ID.");
            } else {
                break;
            }
        }
    }

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
```

```
}

/*
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    The target group specifies how the load balancer forward requests to the
instances
    in the group.
*/
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest .builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to "+asGroupName);

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

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName ) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest.builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

    .map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
```

```
.collect(Collectors.toList());  
  
String availabilityZones = String.join(", ", availabilityZoneNames);  
LaunchTemplateSpecification specification =  
LaunchTemplateSpecification.builder()  
    .launchTemplateName(templateName)  
    .version("$Default")  
    .build();  
  
String[] zones = availabilityZones.split(",");  
CreateAutoScalingGroupRequest groupRequest =  
CreateAutoScalingGroupRequest.builder()  
    .launchTemplate(specification)  
    .availabilityZones(zones)  
    .maxSize(groupSize)  
    .minSize(groupSize)  
    .autoScalingGroupName(autoScalingGroupName)  
    .build();  
  
try {  
    getAutoScalingClient().createAutoScalingGroup(groupRequest);  
  
} catch (AutoScalingException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
System.out.println("Created an EC2 Auto Scaling group named "+  
autoScalingGroupName);  
return zones;  
}  
  
public String getDefaultVPC() {  
    // Define the filter.  
    Filter defaultFilter = Filter.builder()  
        .name("is-default")  
        .values("true")  
        .build();  
  
    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =  
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest.builder()  
    .filters(defaultFilter)  
    .build();  
  
DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
```

```
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        Filter azFilter = Filter.builder()
            .name("availability-zone")
            .values(availabilityZones)
            .build();

        Filter defaultForAZ = Filter.builder()
            .name("default-for-az")
            .values("true")
            .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
            .filters(vpcFilter, azFilter, defaultForAZ)
            .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
            .map(instance -> instance.instanceId())
            .collect(Collectors.toList());
    }
}
```

```
        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
            .name("instance-id")
            .values(instanceId)
            .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest.builder()
            .filters(filter)
            .build();

        DescribeIamInstanceProfileAssociationsResponse response =
getEc2Client().describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile ) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

                software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();
                ListEntitiesForPolicyResponse listEntitiesResponse =
iamClient.listEntitiesForPolicy(listEntitiesRequest);
            }
        }
    }
}
```

```
        if (!listEntitiesResponse.policyGroups().isEmpty())
    || !listEntitiesResponse.policyUsers().isEmpty() || !
listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role
                .build();

            getIAMClient().detachRolePolicy(detachPolicyRequest);
            System.out.println("Policy detached from entities.");
        }

        // Now, you can delete the policy.
        DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .roleName(roleName) // Remove the extra dot here
    .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
```

```
        System.out.println("Role " + roleName + " removed from instance profile  
" + InstanceProfile);  
    }  
  
    // Delete the instance profile after removing all roles  
    DeleteInstanceProfileRequest deleteInstanceProfileRequest =  
DeleteInstanceProfileRequest.builder()  
    .instanceProfileName(InstanceProfile)  
    .build();  
  
    getIAMClient().deleteInstanceProfile(r-  
>r.instanceProfileName(InstanceProfile));  
    System.out.println(InstanceProfile +" Deleted");  
    System.out.println("All roles and policies are deleted.");  
}  
}
```

Crie uma classe que envolva as ações do Elastic Load Balancing.

```
public class LoadBalancer {  
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;  
  
    public ElasticLoadBalancingV2Client getLoadBalancerClient() {  
        if (elasticLoadBalancingV2Client == null) {  
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()  
                .region(Region.US_EAST_1)  
                .build();  
        }  
  
        return elasticLoadBalancingV2Client;  
    }  
  
    // Checks the health of the instances in the target group.  
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {  
        DescribeTargetGroupsRequest targetGroupsRequest =  
DescribeTargetGroupsRequest.builder()  
        .names(targetGroupName)  
        .build();  
  
        DescribeTargetGroupsResponse tgResponse =  
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);  
    }  
}
```

```
    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName){
    DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter.waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}
```

```
// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res =
getLoadBalancerClient().describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient().deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName +" was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true ;
            } else {
                retries-- ;
                System.out.println("Got connection error from load balancer
endpoint, retrying...\"");
                TimeUnit.SECONDS.sleep(15);
            }
        }
    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }
}
```

```
        System.out.println("Status.." + success);
        return success;
    }

/*
    Creates an Elastic Load Balancing target group. The target group specifies
how
    the load balancer forward requests to instances in the group and how
instance
    health is checked.
*/
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
    Creates an Elastic Load Balancing load balancer that uses the specified
subnets
    and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port, String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
```

```
        .collect(Collectors.toList());\n\n        CreateLoadBalancerRequest balancerRequest =\nCreateLoadBalancerRequest.builder()\n            .subnets(subnetIdStrings)\n            .name(lbName)\n            .scheme("internet-facing")\n            .build();\n\n        // Create and wait for the load balancer to become available.\n        CreateLoadBalancerResponse lsResponse =\ngetLoadBalancerClient().createLoadBalancer(balancerRequest);\n        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();\n\n        ElasticLoadBalancingV2Waiter loadBalancerWaiter =\ngetLoadBalancerClient().waiter();\n        DescribeLoadBalancersRequest request =\nDescribeLoadBalancersRequest.builder()\n            .loadBalancerArns(lbARN)\n            .build();\n\n        System.out.println("Waiting for Load Balancer " + lbName + " to become\navailable.");\n        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =\nloadBalancerWaiter.waitUntilLoadBalancerAvailable(request);\n        waiterResponse.matched().response().ifPresent(System.out::println);\n        System.out.println("Load Balancer " + lbName + " is available.");\n\n        // Get the DNS name (endpoint) of the load balancer.\n        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();\n        System.out.println("**** Load Balancer DNS Name: " + lbDNSName);\n\n        // Create a listener for the load balance.\n        Action action = Action.builder()\n            .targetGroupArn(targetGroupARN)\n            .type("forward")\n            .build();\n\n        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()\n\n        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())\n            .defaultActions(action)\n            .port(port)\n            .protocol(protocol)
```

```
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println( "Created listener to forward traffic from load
balancer " + lbName + " to target group " + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName){
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

            getDynamoDbClient().describeTable(describeTableRequest);
        }
    }
}
```

```
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/*
Creates a DynamoDB table to use a recommendation service. The table has a
hash key named 'MediaType' that defines the type of media recommended, such
as
    Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
    forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
    }
}
```

```
.provisionedThroughput()
    ProvisionedThroughput.builder()
        .readCapacityUnits(5L)
        .writeCapacityUnits(5L)
        .build())
    .build();

getDynamoDbClient().createTable(createTableRequest);
System.out.println("Creating table " + tableName + "...");

// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}

}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table->table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
```

```
String mediaType = currentNode.path("MediaType").path("S").asText();
int itemId = currentNode.path("ItemId").path("N").asInt();
String title = currentNode.path("Title").path("S").asText();
String creator = currentNode.path("Creator").path("S").asText();

// Create a Recommendation object and set its properties.
Recommendation rec = new Recommendation();
rec.setMediaType(mediaType);
rec.setItemId(itemId);
rec.setTitle(title);
rec.setCreator(creator);

// Put the item into the DynamoDB table.
mappedTable.putItem(rec); // Add the Recommendation to the list.
}
System.out.println("Added all records to the "+tableName);
}
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName );
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
    }
}
```

```
        .overwrite(true)
        .type("String")
        .build();

    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeElbInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)

- [ReplaceElbInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Começar a usar instâncias

O código de exemplo a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associe um endereço IP elástico à sua instância.
- Conecte-se à sua instância via SSH e, em seguida, limpe os recursos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Exemplos do Amazon ECS usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon ECS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Criar um cluster do

O exemplo de código a seguir mostra como criar um cluster do Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateCluster](#) na Referência AWS SDK for Java 2.x da API.

Criar um serviço

O exemplo de código a seguir mostra como criar um serviço Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateService](#) e Referência AWS SDK for Java 2.x da API.

Excluir um serviço

O exemplo de código a seguir mostra como excluir um serviço do Amazon ECS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DeleteService](#) e Referência AWS SDK for Java 2.x da API.

Descreva os clusters

O exemplo de código a seguir mostra como descrever seus clusters do Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
        }
    }
```

- Para obter detalhes da API, consulte [DescribeClusters](#) Referência AWS SDK for Java 2.x da API.

Descreva as tarefas

O exemplo de código a seguir mostra como descrever suas tarefas do Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [DescribeTasks](#) a Referência AWS SDK for Java 2.x da API.

Listar clusters

O exemplo de código a seguir mostra como listar seus clusters do Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListClusters](#) a Referência AWS SDK for Java 2.x da API.

Atualizar um serviço

O exemplo de código a seguir mostra como atualizar um serviço Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [UpdateService](#) a Referência AWS SDK for Java 2.x da API.

Exemplos de Elastic Load Balancing usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Elastic Load Balancing.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá, Elastic Load Balancing

Os exemplos de código a seguir mostram como começar a usar o Elastic Load Balancing.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public class HelloLoadBalancer {  
  
    public static void main(String[] args){  
        ElasticLoadBalancingV2Client loadBalancingV2Client =  
        ElasticLoadBalancingV2Client.builder()  
            .region(Region.US_EAST_1)  
            .build();  
  
        DescribeLoadBalancersResponse loadBalancersResponse =  
        loadBalancingV2Client.describeLoadBalancers(r->r.pageSize(10));  
        List<LoadBalancer> loadBalancerList = loadBalancersResponse.loadBalancers();  
        for (LoadBalancer lb : loadBalancerList)  
            System.out.println("Load Balancer DNS name = " + lb.dnsName());  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeLoadBalancers](#) Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Crie um ouvinte para um balanceador de carga

O exemplo de código a seguir mostra como criar um ouvinte que encaminha solicitações de um balanceador de carga do ELB para um grupo-alvo.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/*
    Creates an Elastic Load Balancing load balancer that uses the specified
    subnets
    and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port, String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();
```

```
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter.waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println( "Created listener to forward traffic from load
balancer " + lbName + " to target group " + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
}
```

```
        return "";
    }
```

- Para obter detalhes da API, consulte [CreateListener](#) Referência AWS SDK for Java 2.x da API.

Criar um grupo de destino

O exemplo de código a seguir mostra como criar um grupo-alvo do ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/*
     Creates an Elastic Load Balancing target group. The target group specifies
how
     the load balancer forward requests to instances in the group and how
instance
     health is checked.
*/
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
```

```
        String targetGroupArn =
targetGroupResponse.getTargetGroups().get(0).getTargetGroupArn();
        String targetGroup =
targetGroupResponse.getTargetGroups().get(0).getTargetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }
```

- Para obter detalhes da API, consulte [CreateTargetGroup](#) Referência AWS SDK for Java 2.x da API.

Criar um Application Load Balancer

O exemplo de código a seguir mostra como criar um ELB Application Load Balancer.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/*
     Creates an Elastic Load Balancing load balancer that uses the specified
     subnets
     and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port, String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::getSubnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
```

```
.scheme("internet-facing")
.build();

// Create and wait for the load balancer to become available.
CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
    .loadBalancerArns(lbARN)
    .build();

System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter.waitUntilLoadBalancerAvailable(request);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Load Balancer " + lbName + " is available.");

// Get the DNS name (endpoint) of the load balancer.
String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

// Create a listener for the load balance.
Action action = Action.builder()
    .targetGroupArn(targetGroupARN)
    .type("forward")
    .build();

CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

getLoadBalancerClient().createListener(listenerRequest);
System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group " + targetGroupARN);
```

```
// Return the load balancer DNS name.  
return lbDNSName;  
  
} catch (ElasticLoadBalancingV2Exception e) {  
    e.printStackTrace();  
}  
return "";  
}
```

- Para obter detalhes da API, consulte [CreateLoadBalancer](#) Referência AWS SDK for Java 2.x da API.

Excluir um load balancer

O exemplo de código a seguir mostra como excluir um平衡ador de carga ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Deletes a load balancer.  
public void deleteLoadBalancer(String lbName) {  
    try {  
        // Use a waiter to delete the Load Balancer.  
        DescribeLoadBalancersResponse res =  
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));  
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =  
getLoadBalancerClient().waiter();  
        DescribeLoadBalancersRequest request =  
DescribeLoadBalancersRequest.builder()  
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())  
            .build();  
  
        getLoadBalancerClient().deleteLoadBalancer(builder ->  
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
```

```
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =  
loadBalancerWaiter.waitUntilLoadBalancersDeleted(request);  
    waiterResponse.matched().response().ifPresent(System.out::println);  
  
} catch (ElasticLoadBalancingV2Exception e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
}  
System.out.println(lbName + " was deleted.");  
}
```

- Para obter detalhes da API, consulte [DeleteLoadBalancer](#) Referência AWS SDK for Java 2.x da API.

Excluir um grupo de destino

O exemplo de código a seguir mostra como excluir um grupo-alvo do ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Deletes the target group.  
public void deleteTargetGroup(String targetGroupName) {  
    try {  
        DescribeTargetGroupsResponse res =  
getLoadBalancerClient().describeTargetGroups(describe ->  
describe.names(targetGroupName));  
        getLoadBalancerClient().deleteTargetGroup(builder ->  
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));  
    } catch (ElasticLoadBalancingV2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
    }  
    System.out.println(targetGroupName + " was deleted.");  
}
```

- Para obter detalhes da API, consulte [DeleteTargetGroup](#) Referência AWS SDK for Java 2.x da API.

Obtenha a saúde de um grupo-alvo

O exemplo de código a seguir mostra como obter a integridade das instâncias em um grupo-alvo do ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
        DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

    DescribeTargetGroupsResponse tgResponse =
        getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
        DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

    DescribeTargetHealthResponse healthResponse =
        getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}
```

- Para obter detalhes da API, consulte [DescribeTargetHealth](#) Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor Web às solicitações e verificações de integridade atualizando os parâmetros do AWS Systems Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute o cenário interativo em um prompt de comando.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworflow\\  
\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworflow\\  
\\server_startup_script.sh"; // Modify file location.
```

```
public static final String policyFile = "C:\\\\AWS\\\\resworkflow\\\ninstance_policy.json"; // Modify file location.\npublic static final String ssmJSON = "C:\\\\AWS\\\\resworkflow\\\nssm_only_policy.json"; // Modify file location.\npublic static final String failureResponse = "doc-example-resilient-\narchitecture-failure-response";\npublic static final String healthCheck = "doc-example-resilient-architecture-\nhealth-check";\npublic static final String templateName = "doc-example-resilience-template" ;\npublic static final String roleName = "doc-example-resilience-role";\npublic static final String policyName = "doc-example-resilience-pol";\npublic static final String profileName ="doc-example-resilience-prof" ;\n\npublic static final String badCredsProfileName ="doc-example-resilience-prof-\nbc" ;\n\npublic static final String targetGroupName = "doc-example-resilience-tg" ;\npublic static final String autoScalingGroupName = "doc-example-resilience-\ngroup";\npublic static final String lbName = "doc-example-resilience-lb" ;\npublic static final String protocol = "HTTP" ;\npublic static final int port = 80 ;\n\npublic static final String DASHES = new String(new char[80]).replace("\0", "-");\npublic static void main(String[] args) throws IOException, InterruptedException\n{\n    Scanner in = new Scanner(System.in);\n    Database database = new Database();\n    AutoScaler autoScaler = new AutoScaler();\n    LoadBalancer loadBalancer = new LoadBalancer();\n\n    System.out.println(DASHES);\n    System.out.println("Welcome to the demonstration of How to Build and Manage\na Resilient Service!");\n    System.out.println(DASHES);\n\n    System.out.println(DASHES);\n    System.out.println("A - SETUP THE RESOURCES");\n    System.out.println("Press Enter when you're ready to start deploying\nresources.");\n    in.nextLine();\n    deploy(loadBalancer);\n    System.out.println(DASHES);\n    System.out.println(DASHES);
```

```
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println("""
This concludes the demo of how to build and manage a resilient service.
To keep things tidy and to avoid unwanted charges on your account, we can
clean up all AWS resources
that were created for this demo.
""");

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
    Okay, we'll leave the resources intact.
    Don't forget to delete them when you're done with them or you might
incur unexpected charges.
""");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database) throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
```

```
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName );
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println("""
        For this demo, we'll use the AWS SDK for Java (v2) to create several AWS
resources
        to set up a load-balanced web service endpoint and explore some ways to
make it resilient
        against various kinds of failures.

        Some of the resources create by this demo are:
        \t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named
"+tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
        This script starts a Python web server defined in the `server.py` script.
The web server
```

```
    listens to HTTP requests on port 80 and responds to requests to '/' and to
    '/healthcheck'.
```

For demo purposes, this server is run as the root user. In production, the best practice is to

```
    run a web server, such as Apache, with least-privileged credentials.
```

The template also defines an IAM policy that each instance uses to assume a role that grants

```
    permissions to access the DynamoDB recommendation table and Systems Manager
parameters
```

that control the flow of the demo.

```
    """);
```

```
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating an EC2 Auto Scaling group that maintains three
EC2 instances, each in a different Availability Zone.");
System.out.println("**** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println(""""
At this point, you have EC2 instances created. Once each instance starts, it
listens for
HTTP requests. You can see these instances in the console or continue with
the demo.
Press Enter when you're ready to continue.
""");

in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println(""""
Creating an Elastic Load Balancing target group and load balancer. The
target group
defines how the load balancer connects to instances. The load balancer
provides a
single endpoint where clients connect and dispatches requests to instances
in the group.
""");

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with "+subnets.size() +" "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""

```

```
For this example to work, the default security group for your
default VPC must
    allow access from this computer. You can either add it
automatically from this
    example or add it yourself using the AWS Management Console.
    """);

    System.out.println("Do you want to add a rule to security group
"+groupInfo.getGroupName() +" to allow");
    System.out.println("inbound traffic on port "+port +" from your
computer's IP address (y/n) ");
    String ans = in.nextLine();
    if ("y".equalsIgnoreCase(ans)) {
        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
        System.out.println("Security group rule added.");
    } else {
        System.out.println("No security group rule added.");
    }
}

} catch (AutoScalingException e) {
    e.printStackTrace();
}
} else if (wasSuccessful) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" +elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
    System.out.println("you can successfully make a GET request to the load
balancer.");
}

System.out.println("Press Enter when you're ready to continue with the
demo.");
in.nextLine();
}
// A method that controls the demo part of the Java program.
public static void demo( LoadBalancer loadBalancer) throws IOException,
InterruptedException {
```

```
ParameterHelper paramHelper = new ParameterHelper();
System.out.println("Read the ssm_only_policy.json file");
String ssmOnlyPolicy = readFileSync(ssmJSON);
```

```
System.out.println("Resetting parameters to starting values for demo.");
paramHelper.reset();
```

```
System.out.println("")
```

This part of the demonstration shows how to toggle different parts of the system

to create situations where the web service fails, and shows how using a resilient

architecture can keep the web service running in spite of these failures.

At the start, the load balancer endpoint returns recommendations and reports that all targets are healthy.

```
""");
```

```
demoChoices(loadBalancer);
```

```
System.out.println("")
```

The web service running on the EC2 instances gets recommendations by querying a DynamoDB table.

The table name is contained in a Systems Manager parameter named self.param_helper.table.

To simulate a failure of the recommendation service, let's set this parameter to name a non-existent table.

```
""");
```

```
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");
```

```
System.out.println("")
```

\nNow, sending a GET request to the load balancer endpoint returns a failure code. But, the service reports as

healthy to the load balancer because shallow health checks don't check for failure of the recommendation service.

```
""");
```

```
demoChoices(loadBalancer);
```

```
System.out.println("")
```

Instead of failing when the recommendation service fails, the web service can return a static response.

While this is not a perfect solution, it presents the customer with a somewhat better experience than failure.

```
""");
```

```
paramHelper.put(paramHelper.failureResponse, "static");
```

```
System.out.println("")  
Now, sending a GET request to the load balancer endpoint returns a static  
response.  
The service still reports as healthy because health checks are still  
shallow.  
""");  
demoChoices(loadBalancer);  
  
System.out.println("Let's reinstate the recommendation service.");  
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);  
  
System.out.println("")  
Let's also substitute bad credentials for one of the instances in the target  
group so that it can't  
access the DynamoDB recommendation table. We will get an instance id value.  
""");  
  
LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();  
AutoScaler autoScaler = new AutoScaler();  
  
//Create a new instance profile based on badCredsProfileName.  
templateCreator.createInstanceProfile(policyFile, policyName,  
badCredsProfileName, roleName);  
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);  
System.out.println("The bad instance id values used for this demo is  
"+badInstanceId);  
  
String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);  
System.out.println("The association Id value is "+profileAssociationId);  
System.out.println("Replacing the profile for instance " + badInstanceId + "  
with a profile that contains bad credentials");  
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,  
profileAssociationId) ;  
  
System.out.println("")  
Now, sending a GET request to the load balancer endpoint returns either a  
recommendation or a static response,  
depending on which instance is selected by the load balancer.  
""");  
  
demoChoices(loadBalancer);  
  
System.out.println("")
```

```
        Let's implement a deep health check. For this demo, a deep health check
tests whether
        the web service can access the DynamoDB table that it depends on for
recommendations. Note that
        the deep health check is only for ELB routing and not for Auto Scaling
instance health.
        This kind of deep health check is not recommended for Auto Scaling instance
health, because it
        risks accidental termination of all instances in the Auto Scaling group when
a dependent service fails.
        """);

    System.out.println(""""
    By implementing deep health checks, the load balancer can detect when one of
the instances is failing
        and take that instance out of rotation.
    """);

    paramHelper.put(paramHelper.healthCheck, "deep");

    System.out.println(""""
    Now, checking target health indicates that the instance with bad
credentials
        is unhealthy. Note that it might take a minute or two for the load balancer
to detect the unhealthy
        instance. Sending a GET request to the load balancer endpoint always returns
a recommendation, because
        the load balancer takes unhealthy instances out of its rotation.
    """);

    demoChoices(loadBalancer);

    System.out.println(""""
    Because the instances in this demo are controlled by an auto scaler, the
simplest way to fix an unhealthy
        instance is to terminate it and let the auto scaler start a new instance to
replace it.
    """);
    autoScaler.terminateInstance(badInstanceId);

    System.out.println(""""
    Even while the instance is terminating and the new instance is starting,
sending a GET

```

```
request to the web service continues to get a successful recommendation
response because
    the load balancer routes requests to the healthy instances. After the
replacement instance
        starts and reports as healthy, it is included in the load balancing
rotation.
    Note that terminating and replacing an instance typically takes several
minutes, during which time you
        can see the changing health check status until the new instance is running
and healthy.
    """);

    demoChoices(loadBalancer);
    System.out.println("If the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");
    paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

    demoChoices(loadBalancer);
    paramHelper.reset();
}

public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    String[] actions = {
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo."
    };
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("-".repeat(88));
        System.out.println("See the current state of the service by selecting
one of the following choices:");
        for (int i = 0; i < actions.length; i++) {
            System.out.println(i + ": " + actions[i]);
        }

        try {
            System.out.print("\nWhich action would you like to take? ");
            int choice = scanner.nextInt();
            System.out.println("-".repeat(88));

            switch (choice) {
```

```
        case 0 -> {
            System.out.println("Request:\n");
            System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
            CloseableHttpClient httpClient =
HttpClients.createDefault();

            // Create an HTTP GET request to the ELB.
            HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);

            // Display the JSON response
            BufferedReader reader = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
            StringBuilder jsonResponse = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                jsonResponse.append(line);
            }
            reader.close();

            // Print the formatted JSON response.
            System.out.println("Full Response:\n");
            System.out.println(jsonResponse.toString());

            // Close the HTTP client.
            httpClient.close();
        }

        case 1 -> {
            System.out.println("\nChecking the health of load balancer
targets:\n");
            List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
            for (TargetHealthDescription target : health) {
                System.out.printf("\tTarget %s on
port %d is %s%n", target.getTarget().getId(), target.getTarget().getPort(),
target.getTargetHealth().getStateAsString());
            }
        }
    }
}
```

```
        }
        System.out.println("""
            Note that it can take a minute or two for the health
            check to update
            after changes are made.
            """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Crie uma classe que envolva as ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();
}
```

```
getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
When
    * the instance is ready, Systems Manager is used to restart the Python web
server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId) throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification.builder()
    .name(newInstanceProfileName) // Make sure 'newInstanceProfileName' is a
valid IAM Instance Profile name.
    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest.builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure 'profileAssociationId'
is a valid association ID.
    .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
    System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId, newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
```

```
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.instanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
Collections.singletonList("cd / && sudo python3 server.py 80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
    System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress) {
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
    .groupName(secGroupId)
```

```
.cidrIp(ipAddress)
.fromPort(Integer.parseInt(port))
.build();

getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName){
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.instanceProfile().instanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();
    }
}
```

```
// List attached role policies.
ListAttachedRolePoliciesResponse rolesResponse =
getIAMClient().listAttachedRolePolicies(role -> role.roleName(roleName));
    List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
    for (AttachedPolicy attachedPolicy : attachedPolicies) {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(attachedPolicy.policyArn())
            .build();

        getIAMClient().detachRolePolicy(request);
        System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
    }

    getIAMClient().deleteRole(deleteRoleRequest);
    System.out.println("Instance profile and role deleted.");
}

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

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name-
>name.launchTemplateName(templateName));
    System.out.format(templateName + " was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName + " was deleted.");
}
```

```
/*
 Verify the default security group of the specified VPC allows ingress from
this
 computer. This can be done by allowing ingress from this computer's IP
 address. In some situations, such as connecting from a corporate network,
you
 must instead specify a prefix list ID. You can also temporarily open the
port to
 any IP address while running this example. If you do, be sure to remove
public
 access when you're done.

*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
            .filters(filter, filter1)
            .build();

        DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client().describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
```

```
        System.out.println("Found inbound rule: " + ipPermission);
        for (IpRange ipRange : ipPermission.ipRanges()) {
            String cidrIp = ipRange.cidrIp();
            if (cidrIp.startsWith(ipAddress) ||
                cidrIp.equals("0.0.0.0/0")) {
                System.out.println(cidrIp +" is applicable");
                portIsOpen = true;
            }
        }

        if (!ipPermission.prefixListIds().isEmpty()) {
            System.out.println("Prefix lList is applicable");
            portIsOpen = true;
        }

        if (!portIsOpen) {
            System.out.println("The inbound rule does not appear to
be open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or to a prefix
list ID.");
        } else {
            break;
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    The target group specifies how the load balancer forward requests to the
instances
    in the group.
*/
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
```

```
try {
    AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest .builder()
    .autoScalingGroupName(asGroupName)
    .targetGroupARNs(targetGroupARN)
    .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
    System.out.println("Attached load balancer to "+asGroupName);

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

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName ) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest.builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

    String availabilityZones = String.join(",",
availabilityZoneNames);
    LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

    String[] zones = availabilityZones.split(",");
```

```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

    try {
        getAutoScalingClient().createAutoScalingGroup(groupRequest);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Created an EC2 Auto Scaling group named "+
autoScalingGroupName);
    return zones;
}

public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest.builder()
    .filters(defaultFilter)
    .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();
```

```
Filter azFilter = Filter.builder()
    .name("availability-zone")
        .values(availabilityZones)
    .build();

Filter defaultForAZ = Filter.builder()
    .name("default-for-az")
    .values("true")
    .build();

DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
    .filters(vpcFilter, azFilter, defaultForAZ)
    .build();

DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
subnets = response.subnets();
return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
```

```
        Filter filter = Filter.builder()
            .name("instance-id")
            .values(instanceId)
            .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest.builder()
            .filters(filter)
            .build();

        DescribeIamInstanceProfileAssociationsResponse response =
getEc2Client().describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile ) {
    ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
    ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
    for (Policy policy : listPoliciesResponse.policies()) {
        if (policy.policyName().equals(policyName)) {
            // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest.builder()
            .policyArn(policy.arn())
            .build();
            ListEntitiesForPolicyResponse listEntitiesResponse =
iamClient.listEntitiesForPolicy(listEntitiesRequest);
            if (!listEntitiesResponse.policyGroups().isEmpty()
|| !listEntitiesResponse.policyUsers().isEmpty() || !
listEntitiesResponse.policyRoles().isEmpty()) {
                // Detach the policy from any entities it is attached to.
                DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role
                .build();

                getIAMClient().detachRolePolicy(detachPolicyRequest);
```

```
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

    getIAMClient().deletePolicy(deletePolicyRequest);
    System.out.println("Policy deleted successfully.");
    break;
}
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .roleName(roleName) // Remove the extra dot here
    .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
    System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
}

// Delete the instance profile after removing all roles
DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();
```

```
        getIAMClient().deleteInstanceProfile(r-
>r.instanceProfileName(InstanceProfile));
    System.out.println(InstanceProfile +" Deleted");
    System.out.println("All roles and policies are deleted.");
}
}
```

Crie uma classe que envolva as ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
    .names(targetGroupName)
    .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }
}
```

```
// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName){
    DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter.waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res =
getLoadBalancerClient().describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient().deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.out.println(targetGroupName +" was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
    InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true ;
                } else {
                    retries-- ;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }
        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
     * instance
     * health is checked.
     */
}
```

```
public String createTargetGroup(String protocol, int port, String vpcId, String targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
    .healthCheckPath("/healthcheck")
    .healthCheckTimeoutSeconds(5)
    .port(port)
    .vpcId(vpcId)
    .name(targetGroupName)
    .protocol(protocol)
    .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
     Creates an Elastic Load Balancing load balancer that uses the specified
subnets
     and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port, String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
```

```
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter.waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println( "Created listener to forward traffic from load
balancer " + lbName + " to target group " + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName){
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;
        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }
}
```

```
/*
     Creates a DynamoDB table to use a recommendation service. The table has a
     hash key named 'MediaType' that defines the type of media recommended, such
     as
         Book or Movie, and a range key named 'ItemId' that, combined with the
     MediaType,
         forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");
    }
}
```

```
// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}

}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table->table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
    }
}
```

```
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the "+tableName);
}
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName );
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribelamInstanceProfileAssociations](#)
 - [DescribelInstances](#)
 - [DescribeLoadBalancers](#)
 - [DescribeSubnets](#)
 - [DescribeTargetGroups](#)
 - [DescribeTargetHealth](#)
 - [DescribeVpcs](#)
 - [RebootInstances](#)
 - [ReplacelamInstanceProfileAssociation](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

OpenSearch Exemplos de serviços usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with OpenSearch Service.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Crie um domínio

O exemplo de código a seguir mostra como criar um domínio OpenSearch de serviço.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void createNewDomain(OpenSearchClient searchClient, String
domainName) {

    try {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
```

```
.dedicatedMasterCount(3)
.dedicatedMasterType("t2.small.search")
.instanceType("t2.small.search")
.instanceCount(5)
.build();

EBSOptions ebsOptions = EBSOptions.builder()
.ebsEnabled(true)
.volumeSize(10)
.volumeType(VolumeType.GP2)
.build();

NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
.enabled(true)
.build();

CreateDomainRequest domainRequest = CreateDomainRequest.builder()
.domainName(domainName)
.engineVersion("OpenSearch_1.0")
.clusterConfig(clusterConfig)
.ebsOptions(ebsOptions)
.nodeToNodeEncryptionOptions(encryptionOptions)
.build();

System.out.println("Sending domain creation request...");
CreateDomainResponse createResponse =
searchClient.createDomain(domainRequest);
System.out.println("Domain status is
"+createResponse.domainStatus().toString());
System.out.println("Domain Id is
"+createResponse.domainStatus().domainId());

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

- Para obter detalhes da API, consulte [CreateDomain](#) na Referência AWS SDK for Java 2.x da API.

Excluir um domínio

O exemplo de código a seguir mostra como excluir um domínio OpenSearch de serviço.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteSpecificDomain(OpenSearchClient searchClient, String
domainName ) {

    try {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        searchClient.deleteDomain(domainRequest);
        System.out.println(domainName +" was successfully deleted.");

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

- Para obter detalhes da API, consulte [DeleteDomain](#) na Referência AWS SDK for Java 2.x da API.

Listar domínios

O exemplo de código a seguir mostra como listar domínios OpenSearch de serviço.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listAllDomains(OpenSearchClient searchClient){  
  
    try {  
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()  
            .engineType("OpenSearch")  
            .build();  
  
        ListDomainNamesResponse response =  
searchClient.listDomainNames(namesRequest) ;  
        List<DomainInfo> domainInfoList = response.domainNames();  
        for (DomainInfo domain: domainInfoList)  
            System.out.println("Domain name is "+domain.domainName());  
  
    } catch (OpenSearchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [ListDomainNames](#) Referência AWS SDK for Java 2.x da API.

Modificar uma configuração de cluster

O exemplo de código a seguir mostra como modificar a configuração de um cluster de um domínio de OpenSearch serviço.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void updateSpecificDomain(OpenSearchClient searchClient, String
domainName ) {

    try {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        System.out.println("Sending domain update request...");
        UpdateDomainConfigResponse updateResponse =
searchClient.updateDomainConfig(updateDomainConfigRequest);
        System.out.println("Domain update response from Amazon OpenSearch
Service:");
        System.out.println(updateResponse.toString());

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

- Para obter detalhes da API, consulte [UpdateDomainConfig](#) Referência AWS SDK for Java 2.x da API.

EventBridge exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with EventBridge.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá EventBridge

O exemplo de código a seguir mostra como começar a usar o EventBridge.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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 HelloEventBridge {
```

```
public static void main(String[] args) {
    Region region = Region.US_WEST_2;
    EventBridgeClient eventBrClient = EventBridgeClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    listBuses(eventBrClient);
    eventBrClient.close();
}

public static void listBuses( EventBridgeClient eventBrClient) {
    try {
        ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
            .limit(10)
            .build();

        ListEventBusesResponse response =
eventBrClient.listEventBuses(busesRequest);
        List<EventBus> buses = response.eventBuses();
        for (EventBus bus: buses) {
            System.out.println("The name of the event bus is: "+bus.name());
            System.out.println("The ARN of the event bus is: "+bus.arn());
        }
    } catch (EventBridgeException e) {

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

- Para obter detalhes da API, consulte [ListEventBuses](#) Referência AWS SDK for Java 2.x da API

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Adicionar um alvo

O exemplo de código a seguir mostra como adicionar um alvo a um EventBridge evento da Amazon.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Adicione um tópico do Amazon SNS como destino para uma regra.

```
// Add a rule which triggers an SNS target when a file is uploaded to an S3
bucket.

public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn, String topicName, String eventRuleName, String
bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

    eventBrClient.putTargets(request);
    System.out.println("Added event rule "+eventRuleName +" with Amazon SNS
target "+topicName +" for bucket "+bucketName +".");
}
```

Adicione um transformador de entrada a um alvo para uma regra.

```
public static void updateCustomRuleTargetWithTransform(EventBridgeClient  
eventBrClient, String topicArn, String ruleName){  
    String targetId = java.util.UUID.randomUUID().toString();  
    InputTransformer inputTransformer = InputTransformer.builder()  
        .inputTemplate("\\"Notification: sample event was received.\\"")  
        .build();  
  
    Target target = Target.builder()  
        .id(targetId)  
        .arn(topicArn)  
        .inputTransformer(inputTransformer)  
        .build();  
  
    try {  
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()  
            .rule(ruleName)  
            .targets(target)  
            .eventBusName(null)  
            .build();  
  
        eventBrClient.putTargets(targetsRequest);  
    } catch (EventBridgeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [PutTargets](#) a Referência AWS SDK for Java 2.x da API.

Criar uma regra

O exemplo de código a seguir mostra como criar uma EventBridge regra da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie uma regra programada.

```
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);
    }
}
```

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do Amazon Simple Storage Service.

```
// Create a new event rule that triggers when an Amazon S3 object is created in
a bucket.
public static void addEventRule( EventBridgeClient eventBrClient, String
roleArn, String bucketName, String eventRuleName) {
    String pattern = "{\n" +
        "  \"source\": [\"aws.s3\"],\n" +
        "  \"detail-type\": [\"Object Created\"],\n" +
        "  \"detail\": {\n" +
        "    \"bucket\": {\n" +
        "      \"name\": [\""+bucketName+"\"]\n" +
        "    }\n" +
        "  }\n" +
    "}";
}
```

```
try {
    PutRuleRequest ruleRequest = PutRuleRequest.builder()
        .description("Created by using the AWS SDK for Java v2")
        .name(eventRuleName)
        .eventPattern(pattern)
        .roleArn(roleArn)
        .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);
}
}
```

- Para obter detalhes da API, consulte [PutRule](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma regra

O exemplo de código a seguir mostra como excluir uma EventBridge regra da Amazon.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
```

```
}
```

- Para obter detalhes da API, consulte [DeleteRule](#) a Referência AWS SDK for Java 2.x da API.

Descreva uma regra

O exemplo de código a seguir mostra como descrever uma EventBridge regra da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is
"+response.stateAsString());

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

- Para obter detalhes da API, consulte [DescribeRule](#) a Referência AWS SDK for Java 2.x da API.

Desativar uma regra

O exemplo de código a seguir mostra como desativar uma EventBridge regra da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Desative uma regra usando seu nome de regra.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: "+eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: "+eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [DisableRule](#) a Referência AWS SDK for Java 2.x da API.

Habilitar uma regra

O exemplo de código a seguir mostra como habilitar uma EventBridge regra da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Habilite uma regra usando seu nome de regra.

```
public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: "+eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: "+eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [EnableRule](#) a Referência AWS SDK for Java 2.x da API.

Listar nomes de regras para um alvo

O exemplo de código a seguir mostra como listar nomes de EventBridge regras da Amazon para um alvo.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Liste todos os nomes das regras usando o alvo.

```
public static void listTargetRules(EventBridgeClient eventBrClient, String topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
    .targetArn(topicArn)
    .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule:rules) {
        System.out.println("The rule name is "+rule);
    }
}
```

- Para obter detalhes da API, consulte [ListRuleNamesByTarget](#) Referência AWS SDK for Java 2.x da API.

Regras da lista

O exemplo de código a seguir mostra como listar EventBridge as regras da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Habilite uma regra usando seu nome de regra.

```
public static void listRules(EventBridgeClient eventBrClient) {  
    try {  
        ListRulesRequest rulesRequest = ListRulesRequest.builder()  
            .eventBusName("default")  
            .limit(10)  
            .build();  
  
        ListRulesResponse response = eventBrClient.listRules(rulesRequest);  
        List<Rule> rules = response.rules();  
        for (Rule rule : rules) {  
            System.out.println("The rule name is : "+rule.name());  
            System.out.println("The rule description is : "+rule.description());  
            System.out.println("The rule state is : "+rule.stateAsString());  
        }  
    } catch (EventBridgeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [ListRules](#) Referência AWS SDK for Java 2.x da API.

Listar alvos para uma regra

O exemplo de código a seguir mostra como listar EventBridge alvos da Amazon para uma regra.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Liste todos os alvos de uma regra usando o nome da regra.

```
public static void listTargets(EventBridgeClient eventBrClient, String ruleName)  
{
```

```
ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
    .rule(ruleName)
    .build();

ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
List<Target> tagets = res.targets();
for (Target target :tagets) {
    System.out.println("Target ARN: "+target.arn());
}
}
```

- Para obter detalhes da API, consulte [ListTargetsByRule](#) e Referência AWS SDK for Java 2.x da API.

Remover alvos de uma regra

O exemplo de código a seguir mostra como remover EventBridge alvos da Amazon de uma regra.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Remova todos os alvos de uma regra usando o nome da regra.

```
public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
    List<Target> allTargets = response.targets();

    // Get all targets and delete them.
```

```
        for (Target myTarget:allTargets) {  
            RemoveTargetsRequest removeTargetsRequest =  
RemoveTargetsRequest.builder()  
                .rule(eventRuleName)  
                .ids(myTarget.id())  
                .build();  
  
            eventBrClient.removeTargets(removeTargetsRequest);  
            System.out.println("Successfully removed the target");  
        }  
    }  
}
```

- Para obter detalhes da API, consulte [RemoveTargets](#) a Referência AWS SDK for Java 2.x da API.

Enviar eventos

O exemplo de código a seguir mostra como enviar EventBridge eventos da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void triggerCustomRule(EventBridgeClient eventBrClient, String  
email) {  
    String json = "{" +  
        "\"UserEmail\": \""+email+"\", " +  
        "\"Message\": \"This event was generated by example code.\", " +  
        "\"UtcTime\": \"Now.\"" +  
    "}";  
  
    PutEventsRequestEntry entry = PutEventsRequestEntry.builder()  
        .source("ExampleSource")  
        .detail(json)  
        .detailType("ExampleType")  
        .build();
```

```
PutEventsRequest eventsRequest = PutEventsRequest.builder()
    .entries(entry)
    .build();

eventBrClient.putEvents(eventsRequest);
}
```

- Para obter detalhes da API, consulte [PutEvents](#) Referência AWS SDK for Java 2.x da API.

Cenários

Comece com regras e metas

O código de exemplo a seguir mostra como:

- Crie uma regra e adicione um alvo a ela.
- Ative e desative as regras.
- Liste e atualize regras e metas.
- Envie eventos e depois limpe os recursos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**
 * 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:
```

```
*  
* This Java V2 example performs the following tasks with Amazon EventBridge:  
*  
* 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon  
EventBridge.  
* 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events  
enabled.  
* 3. Creates a rule that triggers when an object is uploaded to Amazon S3.  
* 4. Lists rules on the event bus.  
* 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets  
the user subscribe to it.  
* 6. Adds a target to the rule that sends an email to the specified topic.  
* 7. Creates an EventBridge event that sends an email when an Amazon S3 object is  
created.  
* 8. Lists Targets.  
* 9. Lists the rules for the same target.  
* 10. Triggers the rule by uploading a file to the Amazon S3 bucket.  
* 11. Disables a specific rule.  
* 12. Checks and print the state of the rule.  
* 13. Adds a transform to the rule to change the text of the email.  
* 14. Enables a specific rule.  
* 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.  
* 16. Updates the rule to be a custom rule pattern.  
* 17. Sending an event to trigger the rule.  
* 18. Cleans up resources.  
*  
*/  
  
public class EventbridgeMVP {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException, IOException  
{  
        final String usage = "\n" +  
            "Usage:\n" +  
            "      <roleName> <bucketName> <topicName> <eventRuleName>\n\n" +  
            "Where:\n" +  
            "      roleName - The name of the role to create.\n" +  
            "      bucketName - The Amazon Simple Storage Service (Amazon S3) bucket  
name to create.\n" +  
            "      topicName - The name of the Amazon Simple Notification Service  
(Amazon SNS) topic to create.\n" +  
            "      eventRuleName - The Amazon EventBridge rule name to create.\n" ;  
  
        if (args.length != 5) {  
            System.out.println(usage);  
        }  
    }  
}
```

```
        System.exit(1);
    }

    String polJSON = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [{" +
            "\"Effect\": \"Allow\", " +
            "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
            "}, " +
            "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
    "}";

    Scanner sc = new Scanner(System.in);
    String roleName = args[0];
    String bucketName = args[1];
    String topicName = args[2];
    String eventRuleName = args[3];

    Region region = Region.US_EAST_1;
    EventBridgeClient eventBrClient = EventBridgeClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    S3Client s3Client = S3Client.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    Region regionGl = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(regionGl)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    SnsClient snsClient = SnsClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EventBridge example scenario.");
}
```

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

System.out.println(DASHES);
System.out.println("1. Create an AWS Identity and Access Management (IAM)
role to use with Amazon EventBridge.");
String roleArn = createIAMRole(iam, roleName, polJSON);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
if (checkBucket(s3Client, bucketName)) {
    System.out.println("Bucket "+ bucketName +" already exists. Ending this
scenario.");
    System.exit(1);
}

createBucket(s3Client, bucketName);
Thread.sleep(3000);
setBucketNotification(s3Client, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
Thread.sleep(10000);
addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. List rules on the event bus.");
listRules(eventBrClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
String topicArn = createSnsTopic(snsClient, topicName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
```

```
        System.out.println("Enter your email to subscribe to the Amazon SNS topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println("Use the link in the email you received to confirm your subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName, eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule "+eventRuleName +" to use a custom
pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target "+topicArn +".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName );
        } else {
```

```
        System.out.println("The resources will not be cleaned up. ");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
    System.out.println(DASHES);
}

public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client, IamClient iam, String topicArn, String eventRuleName,
String bucketName, String roleName) {
    System.out.println("Removing all targets from the event rule.");
    deleteTargetsFromRule(eventBrClient, eventRuleName);
    deleteRuleByName(eventBrClient, eventRuleName);
    deleteSNSTopic(snsClient, topicArn);
    deleteS3Bucket(s3Client, bucketName);
    deleteRole(iam, roleName);
}

public static void deleteRole(IamClient iam, String roleName) {
    String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
    DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
        .policyArn(policyArn)
        .roleName(roleName)
        .build();

    iam.detachRolePolicy(policyRequest);
    System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

    // Delete the role.
    DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

    iam.deleteRole(roleRequest);
    System.out.println("**** Successfully deleted " + roleName);
}

public static void deleteS3Bucket( S3Client s3Client, String bucketName) {
    // Remove all the objects from the S3 bucket.
    ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
```

```
.bucket(bucketName)
.build();

ListObjectsResponse res = s3Client.listObjects(listObjects);
List<S3Object> objects = res.contents();
ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

for (S3Object myValue : objects) {
    toDelete.add(ObjectIdentifier.builder()
        .key(myValue.key())
        .build());
}

DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
    .bucket(bucketName)
    .delete(Delete.builder()
        .objects(toDelete).build())
    .build();

s3Client.deleteObjects(dor);

// Delete the S3 bucket.
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucketName)
    .build();

s3Client.deleteBucket(deleteBucketRequest);
System.out.println("You have deleted the bucket and the objects");
}

// Delete the SNS topic.
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);
    }
}
```

```
        }

    }

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
    DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
        .name(ruleName)
        .build();

    eventBrClient.deleteRule(ruleRequest);
    System.out.println("Successfully deleted the rule");
}

public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    // First, get all targets that will be deleted.
    ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
        .rule(eventRuleName)
        .build();

    ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
    List<Target> allTargets = response.targets();

    // Get all targets and delete them.
    for (Target myTarget:allTargets) {
        RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
        .rule(eventRuleName)
        .ids(myTarget.id())
        .build();

        eventBrClient.removeTargets(removeTargetsRequest);
        System.out.println("Successfully removed the target");
    }
}

public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
    String json = "{" +
        "\"UserEmail\": \""+email+"\", " +
        "\"Message\": \"This event was generated by example code.\", " +
        "\"UtcTime\": \"Now.\"" +
    "}";
}
```

```
PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
    .source("ExampleSource")
    .detail(json)
    .detailType("ExampleType")
    .build();

PutEventsRequest eventsRequest = PutEventsRequest.builder()
    .entries(entry)
    .build();

eventBrClient.putEvents(eventsRequest);
}

public static void updateCustomRuleTargetWithTransform(EventBridgeClient eventBrClient, String topicArn, String ruleName){
    String targetId = java.util.UUID.randomUUID().toString();
    InputTransformer inputTransformer = InputTransformer.builder()
        .inputTemplate("{\"Notification: sample event was received.\"}")
        .build();

    Target target = Target.builder()
        .id(targetId)
        .arn(topicArn)
        .inputTransformer(inputTransformer)
        .build();

    try {
        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
            .rule(ruleName)
            .targets(target)
            .eventBusName(null)
            .build();

        eventBrClient.putTargets(targetsRequest);
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateToCustomRule(EventBridgeClient eventBrClient, String ruleName) {
    String customEventsPattern = "{" +

```

```
"\"source\": [\"ExampleSource\"], \" +\n    \"\"detail-type\": [\"ExampleType\"]\" +\n    \"\"};\n\n    PutRuleRequest request = PutRuleRequest.builder()\n        .name(ruleName)\n        .description("Custom test rule")\n        .eventPattern(customEventsPattern)\n        .build();\n\n    eventBrClient.putRule(request);\n}\n\n// Update an Amazon S3 object created rule with a transform on the target.\npublic static void updateSnsEventRule(EventBridgeClient eventBrClient, String\ntopicArn, String ruleName){\n    String targetId = java.util.UUID.randomUUID().toString();\n    Map<String, String> myMap = new HashMap<>();\n    myMap.put("bucket", ".$.detail.bucket.name");\n    myMap.put("time", ".$.time");\n\n    InputTransformer inputTransformer = InputTransformer.builder()\n        .inputTemplate(\"Notification: an object was uploaded to bucket\n<bucket> at <time>.\\\"\\\")\n        .inputPathsMap(myMap)\n        .build();\n\n    Target target = Target.builder()\n        .id(targetId)\n        .arn(topicArn)\n        .inputTransformer(inputTransformer)\n        .build();\n\n    try {\n        PutTargetsRequest targetsRequest = PutTargetsRequest.builder()\n            .rule(ruleName)\n            .targets(target)\n            .eventBusName(null)\n            .build();\n\n        eventBrClient.putTargets(targetsRequest);\n    } catch (EventBridgeException e) {\n        System.err.println(e.awsErrorDetails().errorMessage());\n    }\n}
```

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

public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
    try {
        DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
            .name(eventRuleName)
            .build();

        DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
        System.out.println("The state of the rule is
"+response.stateAsString());

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

public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
    try {
        if (!isEnabled) {
            System.out.println("Disabling the rule: "+eventRuleName);
            DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                .name(eventRuleName)
                .build();

            eventBrClient.disableRule(ruleRequest);
        } else {
            System.out.println("Enabling the rule: "+eventRuleName);
            EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                .name(eventRuleName)
                .build();
            eventBrClient.enableRule(ruleRequest);
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
// Create and upload a file to an S3 bucket to trigger an event.
public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
    // Create a unique file name.
    String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
    String fileName = "TextFile"+fileSuffix+".txt";

    File myFile = new File(fileName);
    FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
    BufferedWriter bw = new BufferedWriter(fw);
    bw.write("This is a sample file for testing uploads.");
    bw.close();

    try {
        PutObjectRequest put0b = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        s3Client.putObject(put0b, RequestBody.fromFile(myFile));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
    ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
    .targetArn(topicArn)
    .build();

    ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
    List<String> rules = response.ruleNames();
    for (String rule:rules) {
        System.out.println("The rule name is "+rule);
    }
}
```

```
public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
    ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
        .rule(ruleName)
        .build();

    ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
    List<Target> tagets = res.targets();
    for (Target target :tagets) {
        System.out.println("Target ARN: "+target.arn());
    }
}

// Add a rule which triggers an SNS target when a file is uploaded to an S3
bucket.

public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn, String topicName, String eventRuleName, String
bucketName) {
    String targetID = java.util.UUID.randomUUID().toString();
    Target myTarget = Target.builder()
        .id(targetID)
        .arn(topicArn)
        .build();

    List<Target> targets = new ArrayList<>();
    targets.add(myTarget);
    PutTargetsRequest request = PutTargetsRequest.builder()
        .eventBusName(null)
        .targets(targets)
        .rule(ruleName)
        .build();

    eventBrClient.putTargets(request);
    System.out.println("Added event rule "+eventRuleName +" with Amazon SNS
target "+topicName +" for bucket "+bucketName +".");
}

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);
}

}

public static void listRules(EventBridgeClient eventBrClient) {
try {
    ListRulesRequest rulesRequest = ListRulesRequest.builder()
        .eventBusName("default")
        .limit(10)
        .build();

    ListRulesResponse response = eventBrClient.listRules(rulesRequest);
    List<Rule> rules = response.rules();
    for (Rule rule : rules) {
        System.out.println("The rule name is : "+rule.name());
        System.out.println("The rule description is : "+rule.description());
        System.out.println("The rule state is : "+rule.stateAsString());
    }
}

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

}

public static String createSnsTopic(SnsClient snsClient, String topicName) {
String topicPolicy = "{" +
    "\"Version\": \"2012-10-17\", " +
    "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\", " +
        "\"Effect\": \"Allow\", " +
        "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
        "}, " +
```

```
    "\"Resource\": \"*\",\" +  
    "\"Action\": \"sns:Publish\"" +  
    "}]" +  
    "}";  
  
    Map<String, String> topicAttributes = new HashMap<>();  
    topicAttributes.put("Policy", topicPolicy);  
    CreateTopicRequest topicRequest = CreateTopicRequest.builder()  
        .name(topicName)  
        .attributes(topicAttributes)  
        .build();  
  
    CreateTopicResponse response = snsClient.createTopic(topicRequest);  
    System.out.println("Added topic "+topicName +" for email subscriptions.");  
    return response.topicArn();  
}  
  
// Create a new event rule that triggers when an Amazon S3 object is created in  
a bucket.  
public static void addEventRule( EventBridgeClient eventBrClient, String  
roleArn, String bucketName, String eventRuleName) {  
    String pattern = "{\n" +  
    "  \"source\": [\"aws.s3\"],\n" +  
    "  \"detail-type\": [\"Object Created\"],\n" +  
    "  \"detail\": {\n" +  
    "    \"bucket\": {\n" +  
    "      \"name\": [\""+bucketName+"\"]\n" +  
    "    }\n" +  
    "  }\n" +  
    "}";  
  
    try {  
        PutRuleRequest ruleRequest = PutRuleRequest.builder()  
            .description("Created by using the AWS SDK for Java v2")  
            .name(eventRuleName)  
            .eventPattern(pattern)  
            .roleArn(roleArn)  
            .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);
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                .eventBridgeConfiguration(eventBridgeConfiguration)
                .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest.builder()
                .bucket(bucketName)
                .notificationConfiguration(configuration)
                .skipDestinationValidation(true)
                .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");
        } catch (S3Exception e) {
```

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

    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 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);
            AttachRolePolicyRequest rolePolicyRequest =
            AttachRolePolicyRequest.builder()
                .roleName(rolename)
                .policyArn("arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess")
                .build();
        }
    }
}
```

```
        iam.attachRolePolicy(rolePolicyRequest);
        return response.role().arn();

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [DeleteRule](#)
 - [DescribeRule](#)
 - [DisableRule](#)
 - [EnableRule](#)
 - [ListRuleNamesByTarget](#)
 - [ListRules](#)
 - [ListTargetsByRule](#)
 - [PutEvents](#)
 - [PutRule](#)
 - [PutTargets](#)

Exemplos de previsão usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Forecast.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Crie um conjunto de dados

O exemplo de código a seguir mostra como criar um conjunto de dados Forecast.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String createForecastDataSet(ForecastClient forecast, String name) {
    try {
        Schema schema = Schema.builder()
            .attributes(getSchema())
            .build();

        CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
            .datasetName(name)
            .domain("CUSTOM")
            .datasetType("RELATED_TIME_SERIES")
            .dataFrequency("D")
            .schema(schema)
            .build();

        CreateDatasetResponse response = forecast.createDataset(datasetRequest);
        return response.datasetArn();
    } catch (ForecastException e) {
```

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

    return "";
}

// Create a SchemaAttribute list required to create a data set.
private static List<SchemaAttribute> getSchema() {

    List<SchemaAttribute> schemaList = new ArrayList<>();
    SchemaAttribute att1 = SchemaAttribute.builder()
        .attributeName("item_id")
        .attributeType("string")
        .build();

    SchemaAttribute att2 = SchemaAttribute.builder()
        .attributeName("timestamp")
        .attributeType("timestamp")
        .build();

    SchemaAttribute att3 = SchemaAttribute.builder()
        .attributeName("target_value")
        .attributeType("float")
        .build();

    // Push the SchemaAttribute objects to the List.
    schemaList.add(att1);
    schemaList.add(att2);
    schemaList.add(att3);
    return schemaList;
}
```

- Para obter detalhes da API, consulte [CreateDataset](#) a Referência AWS SDK for Java 2.x da API.

Crie uma previsão

O exemplo de código a seguir mostra como criar uma previsão do Forecast.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String createNewForecast(ForecastClient forecast, String name,
String predictorArn) {

    try {
        CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
            .forecastName(name)
            .predictorArn(predictorArn)
            .build() ;

        CreateForecastResponse response =
forecast.createForecast(forecastRequest);
        return response.forecastArn();

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

- Para obter detalhes da API, consulte [CreateForecast](#) Referência AWS SDK for Java 2.x da API.

Excluir um conjunto de dados

O exemplo de código a seguir mostra como excluir um conjunto de dados do Forecast.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteForecastDataSet(ForecastClient forecast, String myDataSetARN) {  
  
    try {  
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()  
            .datasetArn(myDataSetARN)  
            .build();  
  
        forecast.deleteDataset(deleteRequest);  
        System.out.println("The Data Set was deleted") ;  
  
    } catch (ForecastException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteDataset](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma previsão

O exemplo de código a seguir mostra como excluir uma previsão do Forecast.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteForecastDataSet(ForecastClient forecast, String myDataSetARN) {  
  
    try {  
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()  
            .datasetArn(myDataSetARN)  
            .build();  
  
        forecast.deleteDataset(deleteRequest);  
        System.out.println("The Data Set was deleted");  
  
    } catch (ForecastException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteForecast](#) Referência AWS SDK for Java 2.x da API.

Descreva uma previsão

O exemplo de código a seguir mostra como descrever uma previsão do Forecast.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void describe(ForecastClient forecast, String forecastarn) {  
  
    try {  
        DescribeForecastRequest request = DescribeForecastRequest.builder()  
            .forecastArn(forecastarn)  
            .build();  
    }
```

```
        DescribeForecastResponse response = forecast.describeForecast(request);
        System.out.println("The name of the forecast is "
+response.forecastName());

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

- Para obter detalhes da API, consulte [DescribeForecast](#) Referência AWS SDK for Java 2.x da API.

Listar grupos de conjuntos de dados

O exemplo de código a seguir mostra como listar grupos de conjuntos de dados do Forecast.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listDataGroups(ForecastClient forecast) {

    try {
        ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
            .maxResults(10)
            .build();

        ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
        List<DatasetGroupSummary> groups = response.datasetGroups();
        for (DatasetGroupSummary myGroup : groups) {
            System.out.println("The Data Set name is " +
myGroup.datasetGroupName());
        }
    }
}
```

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

- Para obter detalhes da API, consulte [ListDatasetGroups](#) a Referência AWS SDK for Java 2.x da API.

Listar previsões

O exemplo de código a seguir mostra como listar as previsões do Forecast.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listAllForeCasts(ForecastClient forecast) {

    try {
        ListForecastsRequest request = ListForecastsRequest.builder()
            .maxResults(10)
            .build();

        ListForecastsResponse response = forecast.listForecasts(request);
        List<ForecastSummary> forecasts = response.forecasts();
        for (ForecastSummary forecastSummary : forecasts) {
            System.out.println("The name of the forecast is " +
forecastSummary.forecastName());
        }

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

- Para obter detalhes da API, consulte [ListForecasts](#) na Referência AWS SDK for Java 2.x da API.

AWS Glueexemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS Glue.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um crawler

O exemplo de código a seguir mostra como criar um AWS Glue rastreador.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateCrawler](#) Referência AWS SDK for Java 2.x da API.

Obter um crawler

O exemplo de código a seguir mostra como obter um AWS Glue rastreador.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [GetCrawler](#) a Referência AWS SDK for Java 2.x da API.

Obter um banco de dados do Data Catalog

O exemplo de código a seguir mostra como obter um banco de dados do AWS Glue Data Catalog.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [GetDatabase](#) a Referência AWS SDK for Java 2.x da API.

Obter tabelas de um banco de dados

O exemplo de código a seguir mostra como obter tabelas de um banco de dados no AWS Glue Data Catalog.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [GetTables](#) Referência AWS SDK for Java 2.x da API.

Iniciar um crawler

O exemplo de código a seguir mostra como iniciar um AWS Glue rastreador.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [StartCrawler](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Começar a executar crawlers e trabalhos

O código de exemplo a seguir mostra como:

- Criar um crawler que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados formatado em CSV.
- Listar informações sobre bancos de dados e tabelas no AWS Glue Data Catalog.
- Criar um trabalho para extrair dados em CSV do bucket do S3, transformá-los e carregar a saída formatada em JSON em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte [Tutorial: conceitos básicos do AWS Glue Studio](#).

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 *  
 * 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" +
        "      <iام> <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());
        }
    }

} 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);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateCrawler](#)

- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

HealthImaging exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with HealthImaging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Adicionar uma tag a um recurso

O exemplo de código a seguir mostra como adicionar uma tag a um HealthImaging recurso.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void tagMedicalImagingResource(MedicalImagingClient  
medicalImagingClient,  
                                              String resourceArn,  
                                              Map<String, String> tags) {  
    try {  
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()  
            .resourceArn(resourceArn)  
            .tags(tags)  
            .build();  
  
        medicalImagingClient.tagResource(tagResourceRequest);  
  
        System.out.println("Tags have been added to the resource.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [TagResource](#) a Referência AWS SDK for Java 2.x da API.

Copiar um conjunto de imagens

O exemplo de código a seguir mostra como copiar um conjunto de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String copyMedicalImageSet(MedicalImagingClient  
medicalImagingClient,  
                                         String datastoreId,  
                                         String imageSetId,  
                                         String latestVersionId,  
                                         String destinationImageSetId,  
                                         String destinationVersionId) {  
  
    try {  
        CopySourceImageSetInformation copySourceImageSetInformation =  
CopySourceImageSetInformation.builder()  
            .latestVersionId(latestVersionId)  
            .build();  
  
        CopyImageSetInformation.Builder copyImageSetBuilder =  
CopyImageSetInformation.builder()  
            .sourceImageSet(copySourceImageSetInformation);  
  
        if (destinationImageSetId != null) {  
            copyImageSetBuilder =  
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()  
                .imageSetId(destinationImageSetId)  
                .latestVersionId(destinationVersionId)  
                .build());  
        }  
  
        CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()  
            .datastoreId(datastoreId)  
            .sourceImageSetId(imageSetId)  
            .copyImageSetInformation(copyImageSetBuilder.build())  
            .build();  
  
        CopyImageSetResponse response =  
medicalImagingClient.copyImageSet(copyImageSetRequest);
```

```
        return response.destinationImageSetProperties().imageSetId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- Para obter detalhes da API, consulte [CopyImageSet](#) Referência AWS SDK for Java 2.x da API.

Crie um armazenamento de dados

O exemplo de código a seguir mostra como criar um armazenamento HealthImaging de dados.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
                                                String datastoreName) {
    try {
        CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
            .datastoreName(datastoreName)
            .build();
        CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
        return response.datastoreId();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }

    return "";
}
```

- Para obter detalhes da API, consulte [CreateDatastore](#) a Referência AWS SDK for Java 2.x da API.

Excluir um armazenamento de dados

O exemplo de código a seguir mostra como excluir um armazenamento HealthImaging de dados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
                                                String datastoreID) {
    try {
        DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
            .datastoreId(datastoreID)
            .build();
        medicalImagingClient.deleteDatastore(datastoreRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [DeleteDatastore](#) a Referência AWS SDK for Java 2.x da API.

Excluir um conjunto de imagens

O exemplo de código a seguir mostra como excluir um conjunto de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteMedicalImageSet(MedicalImagingClient  
medicalImagingClient,  
                                         String datastoreId,  
                                         String imagesetId) {  
    try {  
        DeleteImageSetRequest deleteImageSetRequest =  
DeleteImageSetRequest.builder()  
                           .datastoreId(datastoreId)  
                           .imageSetId(imagesetId)  
                           .build();  
  
        medicalImagingClient.deleteImageSet(deleteImageSetRequest);  
  
        System.out.println("The image set was deleted.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteImageSet](#) Referência AWS SDK for Java 2.x da API.

Obtenha uma moldura de imagem

O exemplo de código a seguir mostra como obter uma moldura de imagem.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void getMedicalImageSetFrame(MedicalImagingClient  
medicalImagingClient,  
                                         String destinationPath,  
                                         String datastoreId,  
                                         String imagesetId,  
                                         String imageFrameId) {  
  
    try {  
        GetImageFrameRequest getImageSetMetadataRequest =  
GetImageFrameRequest.builder()  
            .datastoreId(datastoreId)  
            .imageSetId(imagesetId)  
            .imageFrameInformation(ImageFrameInformation.builder()  
                .imageFrameId(imageFrameId)  
                .build())  
            .build();  
        medicalImagingClient.getImageFrame(getImageSetMetadataRequest,  
                                         FileSystems.getDefault().getPath(destinationPath));  
  
        System.out.println("Image frame downloaded to " + destinationPath);  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [GetImageFrame](#) a Referência AWS SDK for Java 2.x da API.

Obtenha propriedades do armazenamento de dados

O exemplo de código a seguir mostra como obter propriedades do armazenamento de HealthImaging dados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient  
medicalImagingClient,  
                                              String datastoreID) {  
    try {  
        GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()  
            .datastoreId(datastoreID)  
            .build();  
        GetDatastoreResponse response =  
medicalImagingClient.getDatastore(datastoreRequest);  
        return response.datastoreProperties();  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    return null;  
}
```

- Para obter detalhes da API, consulte [GetDatastore](#) na Referência AWS SDK for Java 2.x da API.

Obter propriedades do conjunto de imagens

O exemplo de código a seguir mostra como obter as propriedades do conjunto de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient  
medicalImagingClient,  
                                                    String datastoreId,  
                                                    String imagesetId,  
                                                    String versionId) {  
    try {  
        GetImageSetRequest.Builder getImageSetRequestBuilder =  
GetImageSetRequest.builder()  
            .datastoreId(datastoreId)  
            .imageSetId(imagesetId);  
  
        if (versionId != null) {  
            getImageSetRequestBuilder =  
getImageSetRequestBuilder.versionId(versionId);  
        }  
  
        return  
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    return null;  
}
```

- Para obter detalhes da API, consulte [GetImageSet](#) Referência AWS SDK for Java 2.x da API.

Obtenha propriedades de trabalho de importação

O exemplo de código a seguir mostra como obter as propriedades do trabalho de importação.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient  
medicalImagingClient,  
                                         String datastoreId,  
                                         String jobId) {  
  
    try {  
        GetDicomImportJobRequest getDicomImportJobRequest =  
GetDicomImportJobRequest.builder()  
            .datastoreId(datastoreId)  
            .jobId(jobId)  
            .build();  
        GetDicomImportJobResponse response =  
medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);  
        return response.jobProperties();  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    return null;  
}
```

- Para obter detalhes da API, consulte [GetDICOM ImportJob na Referência AWS SDK for Java 2.x da API](#).

Obter metadados para um conjunto de imagens

O exemplo de código a seguir mostra como obter metadados para um conjunto de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient  
medicalImagingClient,  
                                                    String datastoreId,  
                                                    String imagesetId,  
                                                    String versionId) {  
    try {  
        GetImageSetRequest.Builder getImageSetRequestBuilder =  
GetImageSetRequest.builder()  
            .datastoreId(datastoreId)  
            .imageSetId(imagesetId);  
  
        if (versionId != null) {  
            getImageSetRequestBuilder =  
getImageSetRequestBuilder.versionId(versionId);  
        }  
  
        return  
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    return null;  
}
```

- Para obter detalhes da API, consulte [GetImageSetMetadata](#) a Referência AWS SDK for Java 2.x da API.

Importar dados em massa para um armazenamento de dados

O exemplo de código a seguir mostra como importar dados em massa para um armazenamento HealthImaging de dados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String startDicomImportJob(MedicalImagingClient  
medicalImagingClient,  
                                         String jobName,  
                                         String datastoreId,  
                                         String dataAccessRoleArn,  
                                         String inputS3Uri,  
                                         String outputS3Uri) {  
  
    try {  
        StartDicomImportJobRequest startDicomImportJobRequest =  
StartDicomImportJobRequest.builder()  
            .jobName(jobName)  
            .datastoreId(datastoreId)  
            .dataAccessRoleArn(dataAccessRoleArn)  
            .inputS3Uri(inputS3Uri)  
            .outputS3Uri(outputS3Uri)  
            .build();  
        StartDicomImportJobResponse response =  
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);  
        return response.jobId();  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    return "";  
}
```

- Para obter detalhes da API, consulte [StartDICOM ImportJob](#) na AWS SDK for Java 2.x Referência da API.

Listar armazenamentos de dados

O exemplo de código a seguir mostra como listar armazenamentos HealthImaging de dados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
    try {
        ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
            .build();
        ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
        List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

        responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

        return datastoreSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Para obter detalhes da API, consulte [ListDatastores](#) Referência AWS SDK for Java 2.x da API.

Listar versões do conjunto de imagens

O exemplo de código a seguir mostra como listar as versões do conjunto de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
                           String
                           datastoreId,
                           String
                           imagesetId) {
    try {
        ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
        .datastoreId(datastoreId)
        .imageSetId(imagesetId)
        .build();

        ListImageSetVersionsIterable responses =
medicalImagingClient.listImageSetVersionsPaginator(getImageSetRequest);
        List<ImageSetProperties> imageSetProperties = new ArrayList<>();
        responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

        return imageSetProperties;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Para obter detalhes da API, consulte [ListImageSetVersions](#) na Referência AWS SDK for Java 2.x da API.

Listar trabalhos de importação para um armazenamento de dados

O exemplo de código a seguir mostra como listar trabalhos de importação para um armazenamento HealthImaging de dados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
                     String
                     datastoreId) {

    try {
        ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
        .datastoreId(datastoreId)
        .build();
        ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
        return response.jobSummaries();
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return new ArrayList<>();
}
```

- Para obter detalhes da API, consulte [ListDicom ImportJobs](#) na Referência AWS SDK for Java 2.x da API.

Listar as tags de um recurso

O exemplo de código a seguir mostra como listar as tags de um HealthImaging recurso.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
                                String
resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- Para obter detalhes da API, consulte [ListTagsForResource](#) Referência AWS SDK for Java 2.x da API.

Remover uma tag de um recurso

O exemplo de código a seguir mostra como remover uma tag de um HealthImaging recurso.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void untagMedicalImagingResource(MedicalImagingClient  
medicalImagingClient,  
                                              String resourceArn,  
                                              Collection<String> tagKeys) {  
    try {  
        UntagResourceRequest untagResourceRequest =  
UntagResourceRequest.builder()  
                      .resourceArn(resourceArn)  
                      .tagKeys(tagKeys)  
                      .build();  
  
        medicalImagingClient.untagResource(untagResourceRequest);  
  
        System.out.println("Tags have been removed from the resource.");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [UntagResource](#) a Referência AWS SDK for Java 2.x da API.

Pesquisar conjuntos de imagens

O exemplo de código a seguir mostra como pesquisar conjuntos de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

A função utilitária para pesquisar conjuntos de imagens.

```
public static List<ImageSetsMetadataSummary>
searchMedicalImagingImageSets(MedicalImagingClient medicalImagingClient,
String datastoreId, List<SearchFilter> searchFilters) {
    try {
        SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
            .datastoreId(datastoreId)

        .searchCriteria(SearchCriteria.builder().filters(searchFilters).build())
            .build();
        SearchImageSetsIterable responses =
medicalImagingClient.searchImageSetsPaginator(datastoreRequest);
        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

        responses.stream().forEach(response ->
imageSetsMetadataSummaries.addAll(response.imageSetsMetadataSummaries()));

        return imageSetsMetadataSummaries;
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

Caso de uso #1: operador EQUAL.

```
List<SearchFilter> searchFilters =  
Collections.singletonList(SearchFilter.builder()  
    .operator(Operator.EQUAL)  
    .values(SearchByAttributeValue.builder()  
        .dicomPatientId(patientId)  
        .build())  
    .build());  
  
List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =  
searchMedicalImagingImageSets(medicalImagingClient,  
    datastoreId, searchFilters);  
if (imageSetsMetadataSummaries != null) {  
    System.out.println("The image sets for patient " + patientId + " are:\n"  
+ imageSetsMetadataSummaries);  
    System.out.println();  
}
```

Caso de uso #2: BETWEEN operador usando DICOM StudyDate e StudyTime DICOM.

```
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");  
searchFilters = Collections.singletonList(SearchFilter.builder()  
    .operator(Operator.BETWEEN)  
    .values(SearchByAttributeValue.builder()  
  
.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()  
    .dicomStudyDate("19990101")  
    .dicomStudyTime("000000.000")  
    .build())  
    .build(),  
    SearchByAttributeValue.builder()  
  
.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()  
  
.dicomStudyDate((LocalDate.now().format(formatter)))  
    .dicomStudyTime("000000.000")  
    .build())  
    .build())  
    .build());  
  
imageSetsMetadataSummaries =  
searchMedicalImagingImageSets(medicalImagingClient,  
    datastoreId, searchFilters);
```

```
if (imageSetsMetadataSummaries != null) {  
    System.out.println("The image sets searched with BETWEEN operator using  
    DICOMStudyDate and DICOMStudyTime are:\n" +  
        imageSetsMetadataSummaries);  
    System.out.println();  
}
```

Caso de uso #3: operador BETWEEN usando createdAt. Os estudos de tempo foram persistidos anteriormente.

```
searchFilters = Collections.singletonList(SearchFilter.builder()  
    .operator(Operator.BETWEEN)  
    .values(SearchByAttributeValue.builder()  
        .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))  
        .build(),  
        SearchByAttributeValue.builder()  
        .createdAt(Instant.now())  
        .build())  
    .build());  
  
imageSetsMetadataSummaries =  
searchMedicalImagingImageSets(medicalImagingClient,  
    datastoreId, searchFilters);  
if (imageSetsMetadataSummaries != null) {  
    System.out.println("The image sets searched with BETWEEN operator using  
    createdAt are:\n" + imageSetsMetadataSummaries);  
    System.out.println();  
}
```

- Para obter detalhes da API, consulte [SearchImageSets](#) a Referência AWS SDK for Java 2.x da API.

Atualizar metadados do conjunto de imagens

O exemplo de código a seguir mostra como atualizar os metadados do conjunto de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void updateMedicalImageSetMetadata(MedicalImagingClient  
medicalImagingClient,  
                                                 String datastoreId,  
                                                 String imagesetId,  
                                                 String versionId,  
                                                 MetadataUpdates  
metadataUpdates) {  
    try {  
        UpdateImageSetMetadataRequest updateImageSetMetadataRequest =  
UpdateImageSetMetadataRequest.builder()  
            .datastoreId(datastoreId)  
            .imageSetId(imagesetId)  
            .latestVersionId(versionId)  
            .updateImageSetMetadataUpdates(metadataUpdates)  
            .build();  
  
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);  
  
        System.out.println("The image set metadata was updated");  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [UpdateImageSetMetadata](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Marcando um armazenamento de dados

O exemplo de código a seguir mostra como marcar um armazenamento HealthImaging de dados.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Para marcar um armazenamento de dados.

```
final String datastoreArn =
    "arn:aws:medical-imaging:us-
east-1:123456789012: datastore/12345678901234567890123456789012";
TagResource.tagMedicalImagingResource(medicalImagingClient, datastoreArn,
ImmutableMap.of("Deployment", "Development"));
```

A função utilitária para marcar um recurso.

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
                                              String resourceArn,
                                              Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

Para listar as tags de um armazenamento de dados.

```
final String datastoreArn =  
        "arn:aws:medical-imaging:us-  
east-1:123456789012: datastore/12345678901234567890123456789012";  
  
ListTagsForResourceResponse result =  
ListTagsForResource.listMedicalImagingResourceTags(medicalImagingClient,  
datastoreArn);  
if (result != null) {  
    System.out.println("Tags for resource: " + result.tags());  
}
```

A função utilitária para listar as tags de um recurso.

```
public static ListTagsForResourceResponse  
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,  
                                String  
resourceArn) {  
    try {  
        ListTagsForResourceRequest listTagsForResourceRequest =  
ListTagsForResourceRequest.builder()  
            .resourceArn(resourceArn)  
            .build();  
  
        return  
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);  
    } catch (MedicalImagingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    return null;  
}
```

Para desmarcar um armazenamento de dados.

```
final String datastoreArn =
    "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
datastoreArn, Collections.singletonList("Deployment"));
```

A função utilitária para desmarcar um recurso.

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
                                                String resourceArn,
                                                Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
        .resourceArn(resourceArn)
        .tagKeys(tagKeys)
        .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

Marcar um conjunto de imagens

O exemplo de código a seguir mostra como marcar um conjunto de HealthImaging imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Para marcar um conjunto de imagens.

```
final String imageSetArn =
    "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

TagResource.tagMedicalImagingResource(medicalImagingClient, imageSetArn,
ImmutableMap.of("Deployment", "Development"));
```

A função utilitária para marcar um recurso.

```
public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
                                              String resourceArn,
                                              Map<String, String> tags) {
    try {
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(resourceArn)
            .tags(tags)
            .build();

        medicalImagingClient.tagResource(tagResourceRequest);

        System.out.println("Tags have been added to the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Para listar tags para um conjunto de imagens.

```
final String imageSetArn =
    "arn:aws:medical-imaging:us-
east-1:123456789012: datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

ListTagsForResourceResponse result =
ListTagsForResource.listMedicalImagingResourceTags(medicalImagingClient,
imageSetArn);
if (result != null) {
    System.out.println("Tags for resource: " + result.tags());
}
```

A função utilitária para listar as tags de um recurso.

```
public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
                               String
resourceArn) {
    try {
        ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
            .resourceArn(resourceArn)
            .build();

        return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

Para desmarcar um conjunto de imagens.

```
final String imageSetArn =
    "arn:aws:medical-imaging:us-
east-1:123456789012: datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
UntagResource.untagMedicalImagingResource(medicalImagingClient, imageSetArn,
Collections.singletonList("Deployment"));
```

A função utilitária para desmarcar um recurso.

```
public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
                                                String resourceArn,
                                                Collection<String> tagKeys) {
    try {
        UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
        .resourceArn(resourceArn)
        .tagKeys(tagKeys)
        .build();

        medicalImagingClient.untagResource(untagResourceRequest);

        System.out.println("Tags have been removed from the resource.");
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

Exemplos de IAM usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o IAM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Anexar uma política a uma função

O exemplo de código a seguir mostra como anexar uma política do IAM a uma função.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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");  
}
```

- Para obter detalhes da API, consulte [AttachRolePolicy](#) na Referência AWS SDK for Java 2.x da API.

Criar uma política

O exemplo de código a seguir mostra como criar uma política do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "" ;  
}
```

- Para obter detalhes da API, consulte [CreatePolicy](#) a Referência AWS SDK for Java 2.x da API.

Criar uma função

O exemplo de código a seguir mostra como criar uma função do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

    try {
        JSONObject json0bject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json0bject.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);
}
```

- Para obter detalhes da API, consulte [CreateRole](#) a Referência AWS SDK for Java 2.x da API.

Criar um usuário

O exemplo de código a seguir mostra como criar um usuário do IAM.

Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 (AmazonServiceException e) {  
        System.out.println("An error occurred while creating the user: " + e.getMessage());  
    }  
}
```

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

- Para obter detalhes da API, consulte [CreateUser](#) Referência AWS SDK for Java 2.x da API.

Criar uma chave de acesso

O exemplo de código a seguir mostra como criar uma chave de acesso do IAM.

Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
    }
```

- Para obter detalhes da API, consulte [CreateAccessKey](#) Referência AWS SDK for Java 2.x da API.

Criar um alias para uma conta

O exemplo de código a seguir mostra como criar um alias para uma conta do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateAccountAlias](#) Referência AWS SDK for Java 2.x da API.

Excluir uma política

O exemplo de código a seguir mostra como excluir uma política do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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");  
}
```

- Para obter detalhes da API, consulte [DeletePolicy](#) Referência AWS SDK for Java 2.x da API.

Exclusão de um usuário

O exemplo de código a seguir mostra como excluir um usuário do IAM.

⚠ Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

ⓘ Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteUser](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma chave de acesso

O exemplo de código a seguir mostra como excluir uma chave de acesso do IAM.

⚠ Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

ⓘ Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteAccessKey](#) Referência AWS SDK for Java 2.x da API.

Excluir um alias de conta

O exemplo de código a seguir mostra como excluir um alias de conta do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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");  
}
```

- Para obter detalhes da API, consulte [DeleteAccountAlias](#) Referência AWS SDK for Java 2.x da API.

Desanexar uma política de uma função

O exemplo de código a seguir mostra como separar uma política do IAM de uma função.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DetachRolePolicy](#) na Referência AWS SDK for Java 2.x da API.

Listar as chaves de acesso de um usuário

O exemplo de código a seguir mostra como listar as chaves de acesso do IAM de um usuário.

⚠ Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

ⓘ Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [ListAccessKeys](#) Referência AWS SDK for Java 2.x da API.

Listar aliases de conta

O exemplo de código a seguir mostra como listar aliases de contas do IAM.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}

}
```

- Para obter detalhes da API, consulte [ListAccountAliases](#) a Referência AWS SDK for Java 2.x da API.

Listar usuários

O exemplo de código a seguir mostra como listar usuários do IAM.

Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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.permissionsBoundaryAsString());
        }

        if(!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

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

- Para obter detalhes da API, consulte [ListUsers](#) Referência AWS SDK for Java 2.x da API.

Atualizar um usuário

O exemplo de código a seguir mostra como atualizar um usuário do IAM.

⚠ Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

ⓘ Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [UpdateUser](#) Referência AWS SDK for Java 2.x da API.

Atualizar uma chave de acesso

O exemplo de código a seguir mostra como atualizar uma chave de acesso do IAM.

⚠ Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

SDK para Java 2.x

ℹ Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [UpdateAccessKeya](#) Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com平衡amento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor Web às solicitações e verificações de integridade atualizando os parâmetros do AWS Systems Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute o cenário interativo em um prompt de comando.

```
public class Main {  
  
    public static final String fileName = "C:\\AWS\\resworkflow\\recommendations.json"; // Modify file location.  
    public static final String tableName = "doc-example-recommendation-service";  
    public static final String startScript = "C:\\AWS\\resworkflow\\server_startup_script.sh"; // Modify file location.  
    public static final String policyFile = "C:\\AWS\\resworkflow\\instance_policy.json"; // Modify file location.  
    public static final String ssmJSON = "C:\\AWS\\resworkflow\\ssm_only_policy.json"; // Modify file location.  
    public static final String failureResponse = "doc-example-resilient-architecture-failure-response";  
    public static final String healthCheck = "doc-example-resilient-architecture-health-check";  
    public static final String templateName = "doc-example-resilience-template" ;  
    public static final String roleName = "doc-example-resilience-role";  
    public static final String policyName = "doc-example-resilience-pol";  
    public static final String profileName ="doc-example-resilience-prof" ;  
  
    public static final String badCredsProfileName ="doc-example-resilience-profc" ;  
  
    public static final String targetGroupName = "doc-example-resilience-tg" ;  
    public static final String autoScalingGroupName = "doc-example-resilience-group";  
    public static final String lbName = "doc-example-resilience-lb" ;  
    public static final String protocol = "HTTP" ;  
    public static final int port = 80 ;  
  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws IOException, InterruptedException  
{  
        Scanner in = new Scanner(System.in);  
        Database database = new Database();  
        AutoScaler autoScaler = new AutoScaler();  
        LoadBalancer loadBalancer = new LoadBalancer();  
  
        System.out.println(DASHES);  
        System.out.println("Welcome to the demonstration of How to Build and Manage  
a Resilient Service!");  
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("A - SETUP THE RESOURCES");
System.out.println("Press Enter when you're ready to start deploying
resources.");
in.nextLine();
deploy(loadBalancer);
System.out.println(DASHES);
System.out.println(DASHES);
System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
System.out.println("Press Enter when you're ready.");
in.nextLine();
demo(loadBalancer);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("C - DELETE THE RESOURCES");
System.out.println(""""
This concludes the demo of how to build and manage a resilient service.
To keep things tidy and to avoid unwanted charges on your account, we can
clean up all AWS resources
that were created for this demo.
""");

System.out.println("\n Do you want to delete the resources (y/n)? ");
String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

if (userInput.equals("y")) {
    // Delete resources here
    deleteResources(loadBalancer, autoScaler, database);
    System.out.println("Resources deleted.");
} else {
    System.out.println("""
Okay, we'll leave the resources intact.
Don't forget to delete them when you're done with them or you might
incur unexpected charges.
""");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The example has completed. ");
System.out.println("\n Thanks for watching!");
System.out.println(DASHES);
}
```

```
// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database) throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName );
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println("""
        For this demo, we'll use the AWS SDK for Java (v2) to create several AWS
resources
        to set up a load-balanced web service endpoint and explore some ways to
make it resilient
        against various kinds of failures.

        Some of the resources create by this demo are:
        \t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named
"+tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
```

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

System.out.println(DASHES);
System.out.println(""""
Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.

This script starts a Python web server defined in the `server.py` script.
The web server
    listens to HTTP requests on port 80 and responds to requests to '/' and to
    '/healthcheck'.
    For demo purposes, this server is run as the root user. In production, the
best practice is to
        run a web server, such as Apache, with least-privileged credentials.

    The template also defines an IAM policy that each instance uses to assume a
role that grants
        permissions to access the DynamoDB recommendation table and Systems Manager
parameters
        that control the flow of the demo.

""");

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating an EC2 Auto Scaling group that maintains three
EC2 instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
At this point, you have EC2 instances created. Once each instance starts, it
listens for
    HTTP requests. You can see these instances in the console or continue with
the demo.

Press Enter when you're ready to continue.

""");

in.nextLine();
```

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

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
Creating an Elastic Load Balancing target group and load balancer. The
target group
defines how the load balancer connects to instances. The load balancer
provides a
single endpoint where clients connect and dispatches requests to instances
in the group.
""");

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with "+subnets.size()+" "
subnets");
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();
    }
}
```

```
// Print the public IP address.  
System.out.println("Public IP Address: " + ipAddress);  
GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,  
ipAddress);  
if (!groupInfo.isPortOpen()) {  
    System.out.println("")  
    For this example to work, the default security group for your  
default VPC must  
        allow access from this computer. You can either add it  
automatically from this  
        example or add it yourself using the AWS Management Console.  
    "");  
  
    System.out.println("Do you want to add a rule to security group  
"+groupInfo.getGroupName() +" to allow");  
    System.out.println("inbound traffic on port "+port +" from your  
computer's IP address (y/n) ");  
    String ans = in.nextLine();  
    if ("y".equalsIgnoreCase(ans)) {  
        autoScaler.openInboundPort(groupInfo.getGroupName(),  
String.valueOf(port), ipAddress);  
        System.out.println("Security group rule added.");  
    } else {  
        System.out.println("No security group rule added.");  
    }  
}  
  
} catch (AutoScalingException e) {  
    e.printStackTrace();  
}  
} else if (wasSuccessful) {  
    System.out.println("Your load balancer is ready. You can access it by  
browsing to:");  
    System.out.println("\t http://" + elbDnsName);  
} else {  
    System.out.println("Couldn't get a successful response from the load  
balancer endpoint. Troubleshoot by");  
    System.out.println("manually verifying that your VPC and security group  
are configured correctly and that");  
    System.out.println("you can successfully make a GET request to the load  
balancer.");  
}
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }
    // A method that controls the demo part of the Java program.
    public static void demo( LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println("""
            This part of the demonstration shows how to toggle different parts of the
system
            to create situations where the web service fails, and shows how using a
resilient
            architecture can keep the web service running in spite of these failures.

            At the start, the load balancer endpoint returns recommendations and reports
that all targets are healthy.
        """);
        demoChoices(loadBalancer);

        System.out.println("""
            The web service running on the EC2 instances gets recommendations by
querying a DynamoDB table.
            The table name is contained in a Systems Manager parameter named
self.param_helper.table.
            To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.
        """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println("""
            \nNow, sending a GET request to the load balancer endpoint returns a failure
code. But, the service reports as
            healthy to the load balancer because shallow health checks don't check for
failure of the recommendation service.
        """);
        demoChoices(loadBalancer);
```

```
        System.out.println(""\");
        Instead of failing when the recommendation service fails, the web service
        can return a static response.

        While this is not a perfect solution, it presents the customer with a
        somewhat better experience than failure.

        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println(""\";
        Now, sending a GET request to the load balancer endpoint returns a static
        response.

        The service still reports as healthy because health checks are still
        shallow.

        """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println(""\";
        Let's also substitute bad credentials for one of the instances in the target
        group so that it can't
        access the DynamoDB recommendation table. We will get an instance id value.

        """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        //Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
        badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is
        "+badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is "+profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId + "
        with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
        profileAssociationId) ;

        System.out.println("")
```

```
Now, sending a GET request to the load balancer endpoint returns either a recommendation or a static response,  
depending on which instance is selected by the load balancer.  
""");  
  
demoChoices(loadBalancer);  
  
System.out.println("")  
Let's implement a deep health check. For this demo, a deep health check tests whether  
the web service can access the DynamoDB table that it depends on for recommendations. Note that  
the deep health check is only for ELB routing and not for Auto Scaling instance health.  
This kind of deep health check is not recommended for Auto Scaling instance health, because it  
risks accidental termination of all instances in the Auto Scaling group when a dependent service fails.  
""");  
  
System.out.println("")  
By implementing deep health checks, the load balancer can detect when one of the instances is failing  
and take that instance out of rotation.  
""");  
  
paramHelper.put(paramHelper.healthCheck, "deep");  
  
System.out.println("")  
Now, checking target health indicates that the instance with bad credentials  
is unhealthy. Note that it might take a minute or two for the load balancer to detect the unhealthy  
instance. Sending a GET request to the load balancer endpoint always returns a recommendation, because  
the load balancer takes unhealthy instances out of its rotation.  
""");  
  
demoChoices(loadBalancer);  
  
System.out.println("")  
Because the instances in this demo are controlled by an auto scaler, the simplest way to fix an unhealthy
```

```
        instance is to terminate it and let the auto scaler start a new instance to
replace it.
        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
Even while the instance is terminating and the new instance is starting,
sending a GET
request to the web service continues to get a successful recommendation
response because
the load balancer routes requests to the healthy instances. After the
replacement instance
starts and reports as healthy, it is included in the load balancing
rotation.

Note that terminating and replacing an instance typically takes several
minutes, during which time you
can see the changing health check status until the new instance is running
and healthy.
""");

        demoChoices(loadBalancer);
        System.out.println("If the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    String[] actions = {
        "Send a GET request to the load balancer endpoint.",
        "Check the health of load balancer targets.",
        "Go to the next part of the demo."
    };
    Scanner scanner = new Scanner(System.in);

    while (true) {
        System.out.println("-".repeat(88));
        System.out.println("See the current state of the service by selecting
one of the following choices:");
        for (int i = 0; i < actions.length; i++) {
            System.out.println(i + ": " + actions[i]);
        }
    }
}
```

```
}

try {
    System.out.print("\nWhich action would you like to take? ");
    int choice = scanner.nextInt();
    System.out.println("-".repeat(88));

    switch (choice) {
        case 0 -> {
            System.out.println("Request:\n");
            System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
            CloseableHttpClient httpClient =
HttpClients.createDefault();

            // Create an HTTP GET request to the ELB.
            HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);

            // Display the JSON response
            BufferedReader reader = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
            StringBuilder jsonResponse = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                jsonResponse.append(line);
            }
            reader.close();

            // Print the formatted JSON response.
            System.out.println("Full Response:\n");
            System.out.println(jsonResponse.toString());

            // Close the HTTP client.
            httpClient.close();
        }
        case 1 -> {
```

```
        System.out.println("\nChecking the health of load balancer
targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on
port %d is %s%n", target.target().id(), target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println(""""
Note that it can take a minute or two for the health
check to update
after changes are made.
""");
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value between
0-2. Please select again.");
}
}

} catch (java.util.InputMismatchException e) {
    System.out.println("Invalid input. Please select again.");
    scanner.nextLine(); // Clear the input buffer.
}
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}
```

Crie uma classe que envolva as ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {  
  
    private static Ec2Client ec2Client;
```

```
private static AutoScalingClient autoScalingClient;
private static IamClient iamClient;

private static SsmClient ssmClient;

private IamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = IamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance is
 * terminated, it can no longer be accessed.
```

```
/*
public void terminateInstance(String instanceId) {
    TerminateInstanceInAutoScalingGroupRequest terminateInstanceRequest =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/***
 * Replaces the profile associated with a running instance. After the profile is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
When
    * the instance is ready, Systems Manager is used to restart the Python web
server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId) throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification.builder()
        .name(newInstanceProfileName) // Make sure 'newInstanceProfileName' is a
valid IAM Instance Profile name.
        .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest.builder()
        .iamInstanceProfile(iamInstanceProfile)
        .associationId(profileAssociationId) // Make sure 'profileAssociationId'
is a valid association ID.
        .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
}
```

```
        }

        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId, newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.instanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
Collections.singletonList("cd / && sudo python3 server.py 80")))
        .build();

    getSSMClient().sendCommand(sendCommandRequest);
```

```
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName){
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
            .instanceProfileName(profileName)
            .roleName(roleName)
            .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
```

```
.instanceProfileName(profileName)
.build();

getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
System.out.println("Deleted instance profile " + profileName);

DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

// List attached role policies.
ListAttachedRolePoliciesResponse rolesResponse =
getIAMClient().listAttachedRolePolicies(role -> role.roleName(roleName));
List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
for (AttachedPolicy attachedPolicy : attachedPolicies) {
    DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(attachedPolicy.policyArn())
        .build();

    getIAMClient().detachRolePolicy(request);
    System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
}

getIAMClient().deleteRole(deleteRoleRequest);
System.out.println("Instance profile and role deleted.");

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

public void deleteTemplate(String templateName) {
    getEc2Client().deleteLaunchTemplate(name-
>name.launchTemplateName(templateName));
    System.out.format(templateName +" was deleted.");
}

public void deleteAutoScaleGroup(String groupName) {
```

```
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
    System.out.println(groupName +" was deleted.");
}

/*
Verify the default security group of the specified VPC allows ingress from
this
computer. This can be done by allowing ingress from this computer's IP
address. In some situations, such as connecting from a corporate network,
you
must instead specify a prefix list ID. You can also temporarily open the
port to
any IP address while running this example. If you do, be sure to remove
public
access when you're done.

*/
public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
    boolean portIsOpen = false;
    GroupInfo groupInfo = new GroupInfo();
    try {
        Filter filter = Filter.builder()
            .name("group-name")
            .values("default")
            .build();

        Filter filter1 = Filter.builder()
            .name("vpc-id")
            .values(VPC)
            .build();

        DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();
}
```

```
        DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client().describeSecurityGroups(securityGroupsRequest);
        String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
        groupInfo.setGroupName(securityGroup);

        for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
            System.out.println("Found security group: " + secGroup.groupId());

            for (IpPermission ipPermission : secGroup.ipPermissions()) {
                if (ipPermission.fromPort() == port) {
                    System.out.println("Found inbound rule: " + ipPermission);
                    for (IpRange ipRange : ipPermission.ipRanges()) {
                        String cidrIp = ipRange.cidrIp();
                        if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                            System.out.println(cidrIp +" is applicable");
                            portIsOpen = true;
                        }
                    }
                }

                if (!ipPermission.prefixListIds().isEmpty()) {
                    System.out.println("Prefix lList is applicable");
                    portIsOpen = true;
                }
            }

            if (!portIsOpen) {
                System.out.println("The inbound rule does not appear to
be open to either this computer's IP,
                    + " all IP addresses (0.0.0.0/0), or to a prefix
list ID.");
            } else {
                break;
            }
        }
    }

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
```

```
}

/*
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    The target group specifies how the load balancer forward requests to the
instances
    in the group.
*/
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest .builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to "+asGroupName);

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

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName ) {

    // Get availability zones.
    software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest.builder()
    .build();

    DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
    List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

    .map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
```

```
.collect(Collectors.toList());  
  
    String availabilityZones = String.join("", availabilityZoneNames);  
    LaunchTemplateSpecification specification =  
LaunchTemplateSpecification.builder()  
        .launchTemplateName(templateName)  
        .version("$Default")  
        .build();  
  
    String[] zones = availabilityZones.split(",");  
    CreateAutoScalingGroupRequest groupRequest =  
CreateAutoScalingGroupRequest.builder()  
        .launchTemplate(specification)  
        .availabilityZones(zones)  
        .maxSize(groupSize)  
        .minSize(groupSize)  
        .autoScalingGroupName(autoScalingGroupName)  
        .build();  
  
    try {  
        getAutoScalingClient().createAutoScalingGroup(groupRequest);  
  
    } catch (AutoScalingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Created an EC2 Auto Scaling group named "+  
autoScalingGroupName);  
    return zones;  
}  
  
public String getDefaultVPC() {  
    // Define the filter.  
    Filter defaultFilter = Filter.builder()  
        .name("is-default")  
        .values("true")  
        .build();  
  
    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =  
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest.builder()  
        .filters(defaultFilter)  
        .build();  
  
    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
```

```
        return response.vpcs().get(0).vpcId();
    }

// Gets the default subnets in a VPC for a specified list of Availability Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());
}
```

```
        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
            .name("instance-id")
            .values(instanceId)
            .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest.builder()
            .filters(filter)
            .build();

        DescribeIamInstanceProfileAssociationsResponse response =
getEc2Client().describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile ) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

                software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest.builder()
                    .policyArn(policy.arn())
                    .build();
                ListEntitiesForPolicyResponse listEntitiesResponse =
iamClient.listEntitiesForPolicy(listEntitiesRequest);
            }
        }
    }
}
```

```
        if (!listEntitiesResponse.policyGroups().isEmpty())
    || !listEntitiesResponse.policyUsers().isEmpty() || !
listEntitiesResponse.policyRoles().isEmpty()) {
            // Detach the policy from any entities it is attached to.
            DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
                .policyArn(policy.arn())
                .roleName(roleName) // Specify the name of the IAM role
                .build();

            getIAMClient().detachRolePolicy(detachPolicyRequest);
            System.out.println("Policy detached from entities.");
        }

        // Now, you can delete the policy.
        DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

        getIAMClient().deletePolicy(deletePolicyRequest);
        System.out.println("Policy deleted successfully.");
        break;
    }
}

// List the roles associated with the instance profile
ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
    .roleName(roleName)
    .build();

// Detach the roles from the instance profile
ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .roleName(roleName) // Remove the extra dot here
    .build();

    getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
```

```
        System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
    }

    // Delete the instance profile after removing all roles
    DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
    .instanceProfileName(InstanceProfile)
    .build();

    getIAMClient().deleteInstanceProfile(r-
>r.instanceProfileName(InstanceProfile));
    System.out.println(InstanceProfile +" Deleted");
    System.out.println("All roles and policies are deleted.");
}
}
```

Crie uma classe que envolva as ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
    .names(targetGroupName)
    .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);
```

```
    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName){
    DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res =
getLoadBalancerClient().describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter.waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}
```

```
// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res =
getLoadBalancerClient().describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient().deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName +" was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true ;
            } else {
                retries-- ;
                System.out.println("Got connection error from load balancer
endpoint, retrying...\"");
                TimeUnit.SECONDS.sleep(15);
            }
        }
    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }
}
```

```
        System.out.println("Status.." + success);
        return success;
    }

/*
    Creates an Elastic Load Balancing target group. The target group specifies
how
    the load balancer forward requests to instances in the group and how
instance
    health is checked.
*/
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/*
    Creates an Elastic Load Balancing load balancer that uses the specified
subnets
    and forwards requests to the specified target group.
*/
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port, String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
```

```
.collect(Collectors.toList());  
  
        CreateLoadBalancerRequest balancerRequest =  
CreateLoadBalancerRequest.builder()  
            .subnets(subnetIdStrings)  
            .name(lbName)  
            .scheme("internet-facing")  
            .build();  
  
        // Create and wait for the load balancer to become available.  
        CreateLoadBalancerResponse lsResponse =  
getLoadBalancerClient().createLoadBalancer(balancerRequest);  
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();  
  
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =  
getLoadBalancerClient().waiter();  
        DescribeLoadBalancersRequest request =  
DescribeLoadBalancersRequest.builder()  
            .loadBalancerArns(lbARN)  
            .build();  
  
        System.out.println("Waiting for Load Balancer " + lbName + " to become  
available.");  
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =  
loadBalancerWaiter.waitUntilLoadBalancerAvailable(request);  
        waiterResponse.matched().response().ifPresent(System.out::println);  
        System.out.println("Load Balancer " + lbName + " is available.");  
  
        // Get the DNS name (endpoint) of the load balancer.  
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();  
        System.out.println("**** Load Balancer DNS Name: " + lbDNSName);  
  
        // Create a listener for the load balance.  
        Action action = Action.builder()  
            .targetGroupArn(targetGroupARN)  
            .type("forward")  
            .build();  
  
        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()  
.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())  
            .defaultActions(action)  
            .port(port)  
            .protocol(protocol)
```

```
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println( "Created listener to forward traffic from load
balancer " + lbName + " to target group " + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName){
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

            getDynamoDbClient().describeTable(describeTableRequest);
        }
    }
}
```

```
        System.out.println("Table '" + tableName + "' exists.");
        return true;

    } catch (ResourceNotFoundException e) {
        System.out.println("Table '" + tableName + "' does not exist.");
    } catch (DynamoDbException e) {
        System.err.println("Error checking table existence: " + e.getMessage());
    }
    return false;
}

/*
Creates a DynamoDB table to use a recommendation service. The table has a
hash key named 'MediaType' that defines the type of media recommended, such
as
    Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
    forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException {
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
    }
}
```

```
.provisionedThroughput()
    ProvisionedThroughput.builder()
        .readCapacityUnits(5L)
        .writeCapacityUnits(5L)
        .build())
    .build();

getDynamoDbClient().createTable(createTableRequest);
System.out.println("Creating table " + tableName + "...");

// Wait until the Amazon DynamoDB table is created.
DescribeTableRequest tableRequest = DescribeTableRequest.builder()
    .tableName(tableName)
    .build();

WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Table " + tableName + " created.");

// Add records to the table.
populateTable(fileName, tableName);
}

}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table->table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
enhanced client.
public void populateTable(String fileName, String tableName) throws IOException
{
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
```

```
String mediaType = currentNode.path("MediaType").path("S").asText();
int itemId = currentNode.path("ItemId").path("N").asInt();
String title = currentNode.path("Title").path("S").asText();
String creator = currentNode.path("Creator").path("S").asText();

// Create a Recommendation object and set its properties.
Recommendation rec = new Recommendation();
rec.setMediaType(mediaType);
rec.setItemId(itemId);
rec.setTitle(title);
rec.setCreator(creator);

// Put the item into the DynamoDB table.
mappedTable.putItem(rec); // Add the Recommendation to the list.
}
System.out.println("Added all records to the "+tableName);
}
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName );
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
    }
}
```

```
        .overwrite(true)
        .type("String")
        .build();

    ssmClient.putParameter(parameterRequest);
    System.out.printf("Setting demo parameter %s to '%s'.", name, value);
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeElbInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)

- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Criar um usuário e assumir uma função

O exemplo de código a seguir mostra como criar um usuário e assumir uma função.

Warning

Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no [repositório de exemplos de código da AWS](#).

Crie a funções que envolvam ações do usuário do IAM.

```
/*
```

```
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);
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

Trabalhar com a API IAM Policy Builder

O código de exemplo a seguir mostra como:

- Crie políticas do IAM usando a API orientada por objetos.
- Use a API IAM Policy Builder com o serviço do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Os exemplos usam as importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Crie uma política com base no tempo.

```
public String timeBasedPolicyExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addResource(IamResource.ALL)
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.DATE_GREATER_THAN)
                .key("aws:CurrentTime")
                .value("2020-04-01T00:00:00Z")))
        .addCondition(b1 -> b1
            .operator(IamConditionOperator.DATE_LESS_THAN)
            .key("aws:CurrentTime")
            .value("2020-06-30T23:59:59Z")))
    .build();

    // Use an IamPolicyWriter to write out the JSON string to a more readable
    format.

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true)
        .build());
}
```

Crie uma política com várias condições.

```
public String multipleConditionsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addAction("dynamodb:GetItem")
            .addAction("dynamodb:BatchGetItem")
            .addAction("dynamodb:Query")
            .addAction("dynamodb:PutItem")
            .addAction("dynamodb:UpdateItem")
            .addAction("dynamodb:DeleteItem")
            .addAction("dynamodb:BatchWriteItem")
            .addResource("arn:aws:dynamodb:*:*:table/table-name"))

        .addConditions(IamConditionOperator.STRING_EQUALS.addPrefix("ForAllValues:"),  

                      "dynamodb:Attributes",
                      List.of("column-name1", "column-name2", "column-
name3"))
}
```

```

        .addCondition(b1 ->
b1.operator(IamConditionOperator.STRING_EQUALS.addSuffix("IfExists"))
            .key("dynamodb:Select")
            .value("SPECIFIC_ATTRIBUTES")))
        .build();

    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Use entidades principais em uma política.

```

public String specifyPrincipalsExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.DENY)
            .addAction("s3:*")
            .addPrincipal(IamPrincipal.ALL)
            .addResource("arn:aws:s3:::BUCKETNAME/*")
            .addResource("arn:aws:s3:::BUCKETNAME")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.ARN_NOT_EQUALS)
                .key("aws:PrincipalArn")
                .value("arn:aws:iam::444455556666:user/user-name")))
        .build());
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}

```

Permitir o acesso entre contas ao .

```

public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS, "111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.STRING_EQUALS)
                .key("s3:x-amz-acl"))

```

```
        .value("bucket-owner-full-control")))
    .build();
return policy.toJson(IamPolicyWriter.builder()
    .prettyPrint(true).build());
}
```

Crie e carregue uma IamPolicy.

```
public String createAndUploadPolicyExample(IamClient iam, String accountID,
String policyName) {
    // Build the policy.
    IamPolicy policy =
        IamPolicy.builder() // 'version' defaults to "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-east-1:" +
accountID + ":table/exampleTableName")
            .build())
        .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}
```

Baixe e trabalhe com uma IamPolicy.

```
public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName, String newPolicyName) {

    String policyArn = "arn:aws:iam://" + accountID + ":policy/" + policyName;
    GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

    String policyVersion = getPolicyResponse.policy().defaultVersionId();
    GetPolicyVersionResponse getPolicyVersionResponse =
        iam.getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

    // Create an IamPolicy instance from the JSON string returned from IAM.
```

```
        String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
StandardCharsets.UTF_8);
    IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

    /*
     All IamPolicy components are immutable, so use the copy method that creates
     a new instance that
     can be altered in the same method call.

     Add the ability to get an item from DynamoDB as an additional action.
    */
    IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

    // Create a new statement that replaces the original statement.
    IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

    // Upload the new policy. IAM now has both policies.
    iam.createPolicy(r -> r.policyName(newPolicyName)
        .policyDocument(newPolicy.toJson()));

    return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Java 2.x](#).
- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreatePolicy](#)
 - [GetPolicy](#)
 - [GetPolicyVersion](#)

Exemplos do Amazon Keyspaces usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Keyspaces.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá, Amazon Keyspaces

Os exemplos de código a seguir mostram como começar a usar o Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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);
        }
    }
}
```

- Para obter detalhes da API, consulte [ListKeyspaces](#) a Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Crie um keyspace

O exemplo de código a seguir mostra como criar um keyspace do Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateKeyspace](#) a Referência AWS SDK for Java 2.x da API.

Criar uma tabela

O exemplo de código a seguir mostra como criar uma tabela do Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [CreateTable](#) a Referência AWS SDK for Java 2.x da API.

Excluir um keyspace

O exemplo de código a seguir mostra como excluir um keyspace do Amazon Keyspaces.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteKeyspace](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma tabela

O exemplo de código a seguir mostra como excluir uma tabela do Amazon Keyspaces.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteTable](#) a Referência AWS SDK for Java 2.x da API.

Obtenha dados sobre um keyspace

O exemplo de código a seguir mostra como obter dados sobre um keyspace do Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [GetKeyspacea Referência AWS SDK for Java 2.x da API.](#)

Obter dados sobre uma tabela

O exemplo de código a seguir mostra como obter dados sobre uma tabela do Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
        }
    }
```

- Para obter detalhes da API, consulte [GetTablea](#) Referência AWS SDK for Java 2.x da API.

Listar espaços chave

O exemplo de código a seguir mostra como listar os keyspaces do Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [ListKeyspaces](#) a Referência AWS SDK for Java 2.x da API.

Listar tabelas em um keyspace

O exemplo de código a seguir mostra como listar tabelas do Amazon Keyspaces em um keyspace.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListTables](#) a Referência AWS SDK for Java 2.x da API.

Restaurar uma tabela para um determinado momento

O exemplo de código a seguir mostra como restaurar uma tabela do Amazon Keyspaces em um determinado momento.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [RestoreTablea Referência AWS SDK for Java 2.x da API](#).

Atualizar uma tabela

O exemplo de código a seguir mostra como atualizar uma tabela do Amazon Keyspaces.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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)
            .addColumn(def)
            .build();

        keyClient.updateTable(tableRequest);

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

- Para obter detalhes da API, consulte [UpdateTable](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Comece a usar espaços de teclas e tabelas

O código de exemplo a seguir mostra como:

- Crie um espaço de teclas e uma tabela. O esquema da tabela contém dados do filme e tem a point-in-time recuperação ativada.
- Conecte-se ao keyspace usando uma conexão TLS segura com autenticação SigV4.
- Consulte a tabela. Adicione, recupere e atualize dados do filme.
- Atualize a tabela. Adicione uma coluna para acompanhar os filmes assistidos.
- Restaure a tabela ao estado anterior e limpe os recursos.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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)
        .addColumn(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 keyspaceRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspaceRequest);
        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);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateKeyspace](#)

- [CreateTable](#)
- [DeleteKeyspace](#)
- [DeleteTable](#)
- [GetKeyspace](#)
- [GetTable](#)
- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

Exemplos do Kinesis usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Exemplos sem servidor](#)

Ações

Criar um stream do

O exemplo de código a seguir mostra como criar um stream do Kinesis.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [CreateStream](#) na Referência AWS SDK for Java 2.x da API.

Excluir um stream

O exemplo de código a seguir mostra como excluir um stream do Kinesis.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteStream](#) na Referência AWS SDK for Java 2.x da API.

Obtenha dados em lotes de um stream

O exemplo de código a seguir mostra como obter dados em lotes de um stream do Kinesis.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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()));
}
```

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [GetRecords](#)
 - [GetShardIterator](#)

Colocar dados em um stream

O exemplo de código a seguir mostra como colocar dados em um stream do Kinesis.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
}  
}
```

- Para obter detalhes da API, consulte [PutRecord](#) a Referência AWS SDK for Java 2.x da API.

Exemplos sem servidor

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do Kinesis com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;  
import com.amazonaws.services.lambda.runtime.RequestHandler;  
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;  
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;
```

```
public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,  
StreamsEventResponse> {  
  
    @Override  
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {  
  
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new  
ArrayList<>();  
        String curRecordSequenceNumber = "";  
  
        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :  
input.getRecords()) {  
            try {  
                //Process your record  
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();  
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();  
  
            } catch (Exception e) {  
                /* Since we are working with streams, we can return the failed item  
immediately.  
                    Lambda will immediately begin to retry processing from this  
failed item onwards. */  
                batchItemFailures.add(new  
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));  
                return new StreamsEventResponse(batchItemFailures);  
            }  
        }  
  
        return new StreamsEventResponse(batchItemFailures);  
    }  
}
```

AWS KMS exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS KMS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Crie uma concessão para uma chave

O exemplo de código a seguir mostra como criar uma concessão para uma chave KMS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
    }
```

- Para obter detalhes da API, consulte [CreateGrant](#) Referência AWS SDK for Java 2.x da API.

Crie uma chave

O exemplo de código a seguir mostra como criar um AWS KMS key.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateKey](#) Referência AWS SDK for Java 2.x da API.

Crie um alias para uma chave

O exemplo de código a seguir mostra como criar um alias para uma chave de chave KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateAlias](#) Referência AWS SDK for Java 2.x da API.

Descriptografar texto cifrado

O exemplo de código a seguir mostra como descriptografar texto cifrado que foi criptografado por uma chave KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [Decrypt](#) na AWS SDK for Java 2.x Referência da API.

Descreva uma chave

O exemplo de código a seguir mostra como descrever uma chave KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeKey](#) a Referência AWS SDK for Java 2.x da API.

Desativar uma chave

O exemplo de código a seguir mostra como desativar uma chave KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
```

- Para obter detalhes da API, consulte [DisableKey](#) a Referência AWS SDK for Java 2.x da API.

Ativar uma chave

O exemplo de código a seguir mostra como habilitar uma chave KMS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [EnableKey](#) a Referência AWS SDK for Java 2.x da API.

Criptografar texto usando uma chave

O exemplo de código a seguir mostra como criptografar texto usando uma chave KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;  
}
```

- Para obter detalhes da API, consulte [Criptografar](#) na Referência AWS SDK for Java 2.x da API.

Listar aliases para uma chave

O exemplo de código a seguir mostra como listar aliases para uma chave KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListAliases](#) a Referência AWS SDK for Java 2.x da API.

Listar subsídios para uma chave

O exemplo de código a seguir mostra como listar concessões para uma chave KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListGrants](#) na Referência AWS SDK for Java 2.x da API.

Listar chaves

O exemplo de código a seguir mostra como listar as chaves KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListKeys](#) a Referência AWS SDK for Java 2.x da API.

Exemplos de Lambda usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Lambda.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

Ações

Criar uma função

O exemplo de código a seguir mostra como criar uma função Lambda.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateFunction](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma função

O exemplo de código a seguir mostra como excluir uma função Lambda.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteFunction](#) Referência AWS SDK for Java 2.x da API.

Invocar uma função

O exemplo de código a seguir mostra como invocar uma função Lambda.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [Invoke](#), na Referência de APIs do AWS SDK for Java 2.x.

Cenários

Conceitos básicos de funções

O código de exemplo a seguir mostra como:

- Crie um perfil do IAM e uma função do Lambda e carregue o código de manipulador.
- Invoque essa função com um único parâmetro e receba resultados.
- Atualize o código de função e configure usando uma variável de ambiente.
- Invoque a função com novos parâmetros e receba resultados. Exiba o log de execução retornado.
- Liste as funções para sua conta e limpe os recursos.

Para obter mais informações, consulte [Criar uma função do Lambda no console](#).

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/*
 * 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 local file containing the Lambda function's code.\n" +
            "      <role> - The ARN of the IAM role that provides execution permissions for the Lambda function.\n" +
            "      <handler> - The fully qualified class name and method name of the Lambda function.\n" +
            "      <bucketName> - The name of the S3 bucket where the Lambda function's code is stored.\n" +
            "      <key> - The key name of the object in the S3 bucket.\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);
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo upload de um objeto em um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar o tipo de conteúdo do objeto.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando Java.

```
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

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

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();
        }
    }
}
```

```
S3Client s3Client = S3Client.builder().build();
HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

    logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

    return "Ok";
} catch (Exception e) {
    throw new RuntimeException(e);
}
}

private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucket)
        .key(key)
        .build();
    return s3Client.headObject(headObjectRequest);
}
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do Kinesis com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
immediately.
                    Lambda will immediately begin to retry processing from this
failed item onwards. */
                batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse(batchItemFailures);
    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções Lambda que recebem eventos de uma fila SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
    }
}
```

```
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

MediaConvert exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with MediaConvert.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Crie um trabalho de transcodificação

O exemplo de código a seguir mostra como criar uma tarefa de AWS Elemental MediaConvert transcodificação.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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();
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();

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.DI  
  
.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_a  
  
.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;
}

}
```

- Para obter detalhes da API, consulte [CreateJob](#) Referência AWS SDK for Java 2.x da API.

Consegna um trabalho de transcodificação

O exemplo de código a seguir mostra como obter um trabalho de AWS Elemental MediaConvert transcodificação.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [GetJob](#) Referência AWS SDK for Java 2.x da API.

Listar trabalhos de transcodificação

O exemplo de código a seguir mostra como listar trabalhos de AWS Elemental MediaConvert transcodificação.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [ListJobs](#) na Referência AWS SDK for Java 2.x da API.

Exemplos do Migration Hub usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Migration Hub.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Excluir fluxo de progresso

O exemplo de código a seguir mostra como excluir o fluxo de progresso.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteStream(MigrationHubClient migrationClient, String streamName) {  
  
    try {  
        DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =  
DeleteProgressUpdateStreamRequest.builder()  
            .progressUpdateStreamName(streamName)  
            .build();  
  
migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);  
        System.out.println(streamName + " is deleted" );  
  
    } catch(MigrationHubException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteProgressUpdateStream](#) na Referência AWS SDK for Java 2.x da API.

Descrever o status da migração

O exemplo de código a seguir mostra como descrever o status da migração.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void describeApplicationState (MigrationHubClient migrationClient,  
String appId) {  
  
    try {
```

```
        DescribeApplicationStateRequest applicationStateRequest =  
DescribeApplicationStateRequest.builder()  
            .applicationId(appId)  
            .build();  
  
        DescribeApplicationStateResponse applicationStateResponse =  
migrationClient.describeApplicationState(applicationStateRequest);  
        System.out.println("The application status is "  
+applicationStateResponse.applicationStatusAsString() );  
  
    } catch(MigrationHubException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeApplicationState](#) Referência AWS SDK for Java 2.x da API.

Obtenha uma lista de atributos associados a uma migração

O exemplo de código a seguir mostra como obter uma lista de atributos associados a uma migração.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void describeMigTask(MigrationHubClient migrationClient, String  
migrationTask, String progressStream) {  
    try {  
        DescribeMigrationTaskRequest migrationTaskRequestRequest =  
DescribeMigrationTaskRequest.builder()  
            .progressUpdateStream(progressStream)  
            .migrationTaskName(migrationTask)  
            .build();
```

```
        DescribeMigrationTaskResponse migrationTaskResponse =
migrationClient.describeMigrationTask(migrationTaskRequestRequest);
        System.out.println("The name is "+
migrationTaskResponse.migrationTask().migrationTaskName());

    } catch (MigrationHubException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [DescribeMigrationTask](#) Referência AWS SDK for Java 2.x da API.

Listar aplicativos

O exemplo de código a seguir mostra como listar aplicativos.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listApps(MigrationHubClient migrationClient) {

    try{
        ListApplicationStatesRequest applicationStatesRequest =
ListApplicationStatesRequest.builder()
            .maxResults(10)
            .build();

        ListApplicationStatesResponse response =
migrationClient.listApplicationStates(applicationStatesRequest);
        List<ApplicationState> apps = response.applicationStateList();
        for (ApplicationState appState : apps) {
```

```
        System.out.println("App Id is " + appState.applicationId());
        System.out.println("The status is " +
appState.applicationStatus().toString());
    }

} catch(MigrationHubException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- Para obter detalhes da API, consulte [ListApplications](#) a Referência AWS SDK for Java 2.x da API.

Listar artefatos criados

O exemplo de código a seguir mostra como listar artefatos criados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listArtifacts(MigrationHubClient migrationClient) {

    try {
        ListCreatedArtifactsRequest listCreatedArtifactsRequest =
ListCreatedArtifactsRequest.builder()
        .maxResults(10)
        .migrationTaskName("SampleApp5")
        .progressUpdateStream("ProgressSteamB")
        .build();

        ListCreatedArtifactsResponse response =
migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
        List<CreatedArtifact> apps = response.createdArtifactList();
        for (CreatedArtifact artifact : apps) {
```

```
        System.out.println("APp Id is " + artifact.description());
        System.out.println("The name is " + artifact.name());
    }

} catch(MigrationHubException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- Para obter detalhes da API, consulte [ListCreatedArtifacts](#) a Referência AWS SDK for Java 2.x da API.

Listar tarefas de migração

O exemplo de código a seguir mostra como listar as tarefas de migração.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listMigrTasks(MigrationHubClient migrationClient) {

    try{
        ListMigrationTasksRequest listMigrationTasksRequest =
ListMigrationTasksRequest.builder()
        .maxResults(10)
        .build();

        ListMigrationTasksResponse response =
migrationClient.listMigrationTasks(listMigrationTasksRequest);
        List<MigrationTaskSummary> migrationList =
response.migrationTaskSummaryList();
        for (MigrationTaskSummary migration : migrationList) {
            System.out.println("Migration task name is " +
migration.migrationTaskName());
```

```
        System.out.println("The Progress update stream is " +
migration.progressUpdateStream());
    }

} catch(MigrationHubException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- Para obter detalhes da API, consulte [ListMigrationTasks](#) a Referência AWS SDK for Java 2.x da API.

Registrar uma tarefa de migração

O exemplo de código a seguir mostra como registrar uma tarefa de migração.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void importMigrTask(MigrationHubClient migrationClient, String
migrationTask, String progressStream) {

    try {
        CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
CreateProgressUpdateStreamRequest.builder()
            .progressUpdateStreamName(progressStream)
            .dryRun(false)
            .build();

        migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
        ImportMigrationTaskRequest migrationTaskRequest =
ImportMigrationTaskRequest.builder()
            .migrationTaskName(migrationTask)
            .progressUpdateStream(progressStream)
    }
}
```

```
        .dryRun(false)
        .build();

    migrationClient.importMigrationTask(migrationTaskRequest);

} catch(MigrationHubException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- Para obter detalhes da API, consulte [ImportMigrationTaska](#) Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Personalize usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Personalize.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Criar uma tarefa de interface em lote

O exemplo de código a seguir mostra como criar um trabalho de interface em lote do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateBatchInferenceJob](#) Referência AWS SDK for Java 2.x da API.

Criar uma campanha

O exemplo de código a seguir mostra como criar uma campanha Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateCampaign](#) Referência AWS SDK for Java 2.x da API.

Crie um conjunto de dados

O exemplo de código a seguir mostra como criar um conjunto de dados Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateDataset](#) na Referência AWS SDK for Java 2.x da API.

Criar um trabalho de exportação de conjunto de dados

O exemplo de código a seguir mostra como criar um trabalho de exportação de conjunto de dados do Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

```
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Para obter detalhes da API, consulte [CreateDatasetExportJoba](#) Referência AWS SDK for Java 2.x da API.

Crie um grupo de conjuntos de dados

O exemplo de código a seguir mostra como criar um grupo de conjuntos de dados do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

Crie um grupo de conjuntos de dados de domínio.

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateDatasetGroup](#) a Referência AWS SDK for Java 2.x da API.

Criar um trabalho de importação de conjunto de dados

O exemplo de código a seguir mostra como criar um trabalho de importação de conjunto de dados do Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateDatasetImportJob](#) a Referência AWS SDK for Java 2.x da API.

Criar um esquema de domínio

O exemplo de código a seguir mostra como criar um esquema de domínio Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateSchema](#) a Referência AWS SDK for Java 2.x da API.

Crie um filtro

O exemplo de código a seguir mostra como criar um filtro Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateFilter](#) a Referência AWS SDK for Java 2.x da API.

Crie um recomendador

O exemplo de código a seguir mostra como criar um recomendador do Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
```

```
}
```

- Para obter detalhes da API, consulte [CreateRecommender](#) Referência AWS SDK for Java 2.x da API.

Criar um esquema

O exemplo de código a seguir mostra como criar um esquema Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
    }
```

- Para obter detalhes da API, consulte [CreateSchema](#) Referência AWS SDK for Java 2.x da API.

Crie uma solução

O exemplo de código a seguir mostra como criar uma solução Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateSolution](#) na Referência AWS SDK for Java 2.x da API.

Crie uma versão da solução

O exemplo de código a seguir mostra como criar uma solução Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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();
            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 "";
}
```

- Para obter detalhes da API, consulte [CreateSolutionVersion](#) na Referência AWS SDK for Java 2.x da API.

Crie um rastreador de eventos

O exemplo de código a seguir mostra como criar um rastreador de eventos Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

- Para obter detalhes da API, consulte [CreateEventTracker](#) Referência AWS SDK for Java 2.x da API.

Excluir uma campanha

O exemplo de código a seguir mostra como excluir uma campanha no Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteCampaign](#) Referência AWS SDK for Java 2.x da API.

Excluir uma solução

O exemplo de código a seguir mostra como excluir uma solução no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteSolution](#) na Referência AWS SDK for Java 2.x da API.

Excluir um rastreador de eventos

O exemplo de código a seguir mostra como excluir um rastreador de eventos no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteEventTracker](#) a Referência AWS SDK for Java 2.x da API.

Descreva uma campanha

O exemplo de código a seguir mostra como descrever uma campanha no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeCampaign](#) na Referência AWS SDK for Java 2.x da API.

Descreva uma receita

O exemplo de código a seguir mostra como descrever uma receita no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeRecipe](#) Referência AWS SDK for Java 2.x da API.

Descreva uma solução

O exemplo de código a seguir mostra como descrever uma solução no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeSolution](#) na Referência AWS SDK for Java 2.x da API.

Listar campanhas

O exemplo de código a seguir mostra como listar campanhas no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListCampaigns](#) a Referência AWS SDK for Java 2.x da API.

Listar grupos de conjuntos de dados

O exemplo de código a seguir mostra como listar grupos de conjuntos de dados no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListDatasetGroups](#) a Referência AWS SDK for Java 2.x da API.

Listar receitas

O exemplo de código a seguir mostra como listar receitas no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListRecipes](#) a Referência AWS SDK for Java 2.x da API.

Listar soluções

O exemplo de código a seguir mostra como listar soluções no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListSolutions](#) Referência AWS SDK for Java 2.x da API.

Atualizar uma campanha

O exemplo de código a seguir mostra como atualizar uma campanha Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
    }
```

- Para obter detalhes da API, consulte [UpdateCampaign](#) Referência AWS SDK for Java 2.x da API.

Exemplos de eventos do Amazon Personalize usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Personalize Events.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Importe dados de eventos de interação em tempo real

O exemplo de código a seguir mostra como importar dados de eventos de interação em tempo real para o Amazon Personalize Events.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

```
}
```

- Para obter detalhes da API, consulte [PutEvents](#) Referência AWS SDK for Java 2.x da API.

Importar incrementalmente um usuário

O exemplo de código a seguir mostra como importar incrementalmente um usuário para o Amazon Personalize Events Events Events.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;  
}
```

- Para obter detalhes da API, consulte [PutUsers](#) Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Personalize Runtime usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Personalize Runtime.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Obtenha recomendações (grupo de conjuntos de dados personalizados)

O exemplo de código a seguir mostra como obter recomendações classificadas do Amazon Personalize Runtime Runtime.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

- Para obter detalhes da API, consulte [GetPersonalizedRanking](#) a Referência AWS SDK for Java 2.x da API.

Receba recomendações de um recomendador (grupo de conjuntos de dados de domínio)

O exemplo de código a seguir mostra como obter recomendações do Amazon Personalize Runtime Runtime.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Obtenha uma lista dos itens recomendados.

```
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);
    }
}
```

Obtenha uma lista de itens recomendados de um recomendador criado em um grupo de conjuntos de dados de domínio.

```
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 um filtro ao solicitar recomendações.

```
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);
```

```
}
```

```
}
```

- Para obter detalhes da API, consulte [GetRecommendations](#) na Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Pinpoint usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto.

Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Criar uma campanha

O exemplo de código a seguir mostra como criar uma campanha.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie uma campanha.

```
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;
    }
}
```

- Para obter detalhes da API, consulte [CreateCampaign](#) Referência AWS SDK for Java 2.x da API.

Criar um segmento

O exemplo de código a seguir mostra como criar um segmento.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;  
}
```

- Para obter detalhes da API, consulte [CreateSegment](#) Referência AWS SDK for Java 2.x da API.

Criar uma aplicação

O exemplo de código a seguir mostra como criar um aplicativo.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateApp](#) Referência AWS SDK for Java 2.x da API.

Excluir uma aplicação

O exemplo de código a seguir mostra como excluir um aplicativo.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Exclua um aplicativo.

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteApp](#) Referência AWS SDK for Java 2.x da API.

Excluir um endpoint

O exemplo de código a seguir mostra como excluir um endpoint.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Exclua um endpoint.

```
public static void deletePinEndpoint(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");
}
```

- Para obter detalhes da API, consulte [DeleteEndpoint](#) Referência AWS SDK for Java 2.x da API.

Exportar um endpoint

O exemplo de código a seguir mostra como exportar um endpoint.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Exporte um 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);
}
}
```

- Para obter detalhes da API, consulte [CreateExportJob](#) Referência AWS SDK for Java 2.x da API.

Obtenha endpoints

O exemplo de código a seguir mostra como obter endpoints.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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");
}
```

- Para obter detalhes da API, consulte [GetEndpoint](#) Referência AWS SDK for Java 2.x da API.

Importar um segmento

O exemplo de código a seguir mostra como importar um segmento.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Importe um segmento.

```
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;
}
```

- Para obter detalhes da API, consulte [CreateImportJob](#) a Referência AWS SDK for Java 2.x da API.

Listar endpoints

O exemplo de código a seguir mostra como listar endpoints.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [GetUserEndpoints](#) Referência AWS SDK for Java 2.x da API.

Listar segmentos

O exemplo de código a seguir mostra como listar segmentos.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar segmentos.

```
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("Segment " + segment.id() + " " + segment.name()  
+ " " + segment.lastModifiedDate());  
        }  
  
    } catch ( PinpointException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [GetSegments](#) Referência AWS SDK for Java 2.x da API.

Envie e-mails e mensagens de texto

O exemplo de código a seguir mostra como enviar mensagens de texto e e-mail com o Amazon Pinpoint.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Envie uma mensagem de e-mail.

```
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);
}
}
```

Envie uma mensagem SMS.

```
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);
}
}
```

Envie mensagens SMS em lote.

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [SendMessages](#) Referência AWS SDK for Java 2.x da API.

Atualizar um endpoint

O exemplo de código a seguir mostra como atualizar um endpoint.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

- Para obter detalhes da API, consulte [UpdateEndpoint](#) Referência AWS SDK for Java 2.x da API.

Atualizar canais

O exemplo de código a seguir mostra como atualizar canais.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [GetSmsChannel](#) Referência AWS SDK for Java 2.x da API.

Exemplos de API de voz e SMS do Amazon Pinpoint usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Pinpoint SMS and Voice API.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Envie uma mensagem de voz com o Amazon Pinpoint SMS and Voice API

O exemplo de código a seguir mostra como enviar uma mensagem de voz com o Amazon Pinpoint SMS and Voice API.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [SendVoiceMessage](#) Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Polly usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Obtenha vozes disponíveis para síntese

O exemplo de código a seguir mostra como disponibilizar vozes do Amazon Polly para síntese.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void describeVoice(PollyClient polly) {  
  
    try {  
        DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()  
            .languageCode("en-US")  
            .build();  
  
        DescribeVoicesResponse enUsVoicesResult =  
polly.describeVoices(voicesRequest);  
        List<Voice> voices = enUsVoicesResult.voices();  
        for (Voice myVoice: voices) {  
            System.out.println("The ID of the voice is " +myVoice.id());  
            System.out.println("The gender of the voice is " +  
myVoice.gender());  
        }  
  
    } catch (PollyException e) {  
        System.err.println("Exception caught: " + e);  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeVoices](#) Referência AWS SDK for Java 2.x da API.

Listar léxicos de pronúncia

O exemplo de código a seguir mostra como listar os léxicos de pronúncia do Amazon Polly.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listLexicons(PollyClient client) {
```

```
try {
    ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
        .build();

    ListLexiconsResponse listLexiconsResult =
    client.listLexicons(listLexiconsRequest);
    List<LexiconDescription> lexiconDescription =
    listLexiconsResult.lexicons();
    for (LexiconDescription lexDescription : lexiconDescription) {
        System.out.println("The name of the Lexicon is " +
lexDescription.name());
    }

} catch (PollyException e) {
    System.err.println("Exception caught: " + e);
    System.exit(1);
}
}
```

- Para obter detalhes da API, consulte [ListLexicons](#) Referência AWS SDK for Java 2.x da API.

Sintetizar fala a partir de texto

O exemplo de código a seguir mostra como sintetizar fala a partir de texto com o Amazon Polly.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void talkPolly(PollyClient polly) {

    try {
        DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
        .engine("standard")
        .build();
```

```
        DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
        Voice voice = describeVoicesResult.voices().get(26);
        InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
        AdvancedPlayer player = new AdvancedPlayer(stream,
javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
        player.setPlayBackListener(new PlaybackListener() {

            public void playbackStarted(PlaybackEvent evt) {
                System.out.println("Playback started");
                System.out.println(SAMPLE);
            }

            public void playbackFinished(PlaybackEvent evt) {
                System.out.println("Playback finished");
            }
        });

        // play it!
        player.play();

    } catch (PollyException | JavaLayerException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static InputStream synthesize(PollyClient polly, String text, Voice
voice, OutputFormat format) throws IOException {

    SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
        .text(text)
        .voiceId(voice.id())
        .outputFormat(format)
        .build();

    ResponseInputStream<SynthesizeSpeechResponse> synthRes =
polly.synthesizeSpeech(synthReq);
    return synthRes;
}
```

- Para obter detalhes da API, consulte [SynthesizeSpeech](#) na Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon RDS usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon RDS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar uma instância de banco de dados

O exemplo de código a seguir mostra como criar uma instância de banco de dados Amazon RDS e esperar que ela fique disponível.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public class CreateDBInstance {
```

```
public static long sleepTime = 20;
public static void main(String[] args) {
    final String usage = "\n" +
        "Usage:\n" +
        "  <dbInstanceIdentifier> <dbName> <secretName>\n\n" +
        "Where:\n" +
        "  dbInstanceIdentifier - The database instance identifier.\n" +
        "  dbName - The database name.\n" +
        "  secretName - The name of the AWS Secrets Manager secret that
contains the database credentials.\n" ;

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

    String dbInstanceIdentifier = args[0];
    String dbName = args[1];
    String secretName = args[2];
    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

    createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword()) ;
    waitForInstanceReady(rdsClient, dbInstanceIdentifier) ;
    rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

private static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
```

```
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
                                              String dbInstanceIdentifier,
                                              String dbName,
                                              String userName,
                                              String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.m4.large")
                .engineVersion("8.0")
                .storageType("standard")
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

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

    // Loop until the cluster is ready.
    while (!instanceReady) {
        DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
        List<DBInstance> instanceList = response.dbInstances();
        for (DBInstance instance : instanceList) {
            instanceReadyStr = instance.dbInstanceState();
            if (instanceReadyStr.contains("available"))
                instanceReady = true;
            else {
                System.out.print(".");
                Thread.sleep(sleepTime * 1000);
            }
        }
    }
    System.out.println("Database instance is available!");

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

- Para ter detalhes da API, consulte [CreateDBInstance](#) na Referência de API do AWS SDK for Java 2.x.

Criar um grupo de parâmetros de banco de dados

O exemplo de código a seguir mostra como criar um grupo de parâmetros de banco de dados do Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CreateDB ParameterGroup na Referência AWS SDK for Java 2.x](#) da API.

Criar um snapshot de uma instância de banco de dados

O exemplo de código a seguir mostra como criar um snapshot de uma instância de banco de dados Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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);
    }
}
```

- Para ter detalhes da API, consulte [CreateDBSnapshot](#) na Referência de API do AWS SDK for Java 2.x.

Criar um token de autenticação

O exemplo de código a seguir mostra como criar um token de autenticação para autenticação do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Use a [RdsUtilities](#) classe para gerar um token de autenticação.

```
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 "";
}
```

- Para obter detalhes da API, consulte [GenerateRds AuthToken na Referência AWS SDK for Java 2.x](#) da API.

Excluir uma instância de banco de dados

O exemplo de código a seguir mostra como excluir uma instância de banco de dados Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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().dbInstanceState());}

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

- Para ter detalhes da API, consulte [DeleteDBInstance](#) na Referência de API do AWS SDK for Java 2.x.

Excluir um grupo de parâmetros de banco de dados

O exemplo de código a seguir mostra como excluir um grupo de parâmetros do banco de dados Amazon RDS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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();
    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);
}
}
```

- Para obter detalhes da API, consulte [DeleteDB ParameterGroup](#) na AWS SDK for Java 2.x Referência da API.

Descrever instâncias de banco de dados

O exemplo de código a seguir mostra como descrever as instâncias de banco de dados do Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para ter detalhes da API, consulte [DescribeDBInstances](#) na Referência de API do AWS SDK for Java 2.x.

Descrever grupos de parâmetros de banco de dados

O exemplo de código a seguir mostra como descrever grupos de parâmetros de banco de dados do Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeDB ParameterGroups em Referência AWS SDK for Java 2.x](#) da API.

Descrever as versões de mecanismo de banco de dados

O exemplo de código a seguir mostra como descrever as versões do mecanismo de banco de dados do Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 engine0b: engines) {  
            System.out.println("The name of the DB parameter group family for  
the database engine is "+engine0b.dbParameterGroupFamily());  
            System.out.println("The name of the database engine  
"+engine0b.engine());  
            System.out.println("The version number of the database engine  
"+engine0b.engineVersion());  
        }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeDB EngineVersions em Referência AWS SDK for Java 2.x](#) da API.

Descrever as opções de instâncias de banco de dados

O exemplo de código a seguir mostra como descrever as opções para instâncias de banco de dados do Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeOrderableDB InstanceOptions](#) em Referência de AWS SDK for Java 2.x API.

Descrever os parâmetros em um grupo de parâmetros de banco de dados

O exemplo de código a seguir mostra como descrever parâmetros em um grupo de parâmetros de banco de dados do Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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_increm " ) ==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);
}
}
```

- Para ter detalhes da API, consulte [DescribeDBParameters](#) na Referência de API do AWS SDK for Java 2.x.

Modificar uma instância de banco de dados

O exemplo de código a seguir mostra como modificar uma instância de banco de dados Amazon RDS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void updateInstance(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);
}
}
```

- Para ter detalhes da API, consulte [ModifyDBInstance](#) na Referência de API do AWS SDK for Java 2.x.

Reiniciar uma instância de banco de dados

Os exemplos de código a seguir mostram como reiniciar uma instância de banco de dados do Amazon RDS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para ter detalhes da API, consulte [RebootDBInstance](#) na Referência de API do AWS SDK for Java 2.x.

Recuperar atributos

O exemplo de código a seguir mostra como recuperar atributos que pertencem a uma conta do Amazon RDS.

SDK para Java 2.x



Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DescribeAccountAttributes](#) a Referência AWS SDK for Java 2.x da API.

Atualizar os parâmetros em um grupo de parâmetros de banco de dados

O exemplo de código a seguir mostra como atualizar parâmetros em um grupo de parâmetros de banco de dados do Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Modify auto_increment_offset and auto_increment_incremet 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);
}
}
```

- Para obter detalhes da API, consulte [ModifyDB ParameterGroup na Referência AWS SDK for Java 2.x](#) da API.

Cenários

Começar a usar instâncias de banco de dados

O código de exemplo a seguir mostra como:

- Crie um grupo de parâmetros de banco de dados e defina os valores dos parâmetros.
- Crie uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Crie um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute várias operações.

```
/**  
 * 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 requires an AWS Secrets Manager secret that contains the database  
 credentials. If you do not create a  
 * secret, this example will not work. For details, see:  
 *  
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-  
 services-use-secrets\_RS.html  
 *  
 * This Java example performs these 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> <dbSnapshotIdentifier> <secretName>\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"+
        "    dbSnapshotIdentifier - The snapshot identifier. \n" +
        "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\";\n\n

if (args.length != 6) {
    System.out.println(usage);
    System.exit(1);
}

String dbGroupName = args[0];
String dbParameterGroupFamily = args[1];
String dbInstanceIdentifier = args[2];
String dbName = args[3];
String dbSnapshotIdentifier = args[4];
String secretName = args[5];

Gson gson = new Gson();
User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
String masterUsername = user.getUsername();
String masterUserPassword = user.getPassword();

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();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

public static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();
}
```

```
        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // 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();
            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().dbInstanceState());
    } 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.dbInstanceState();
                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().dbInstanceState());  
        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 engine0b: engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "+engine0b.dbParameterGroupFamily());
                System.out.println("The name of the database engine
"+engine0b.engine());
            }
        }
    }
}
```

```
        System.out.println("The version number of the database engine  
"+engine0b.engineVersion());  
    }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}  
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateDBInstance](#)
 - [Criado B ParameterGroup](#)
 - [CreateDBSnapshot](#)
 - [DeleteDBInstance](#)
 - [Banco de dados excluído ParameterGroup](#)
 - [DB descrito EngineVersions](#)
 - [DescribeDBInstances](#)
 - [DB descrito ParameterGroups](#)
 - [DescribeDBParameters](#)
 - [DescribeDBSnapshots](#)
 - [DescribeOrderableDB InstanceOptions](#)
 - [Modificar banco de dados ParameterGroup](#)

Exemplos do Amazon Redshift usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Redshift.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Criar um cluster do

O exemplo de código a seguir mostra como criar um cluster do Amazon Redshift.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Criar um cluster do

```
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("dc2.large")
            .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);
    }
}
```

- Para obter detalhes da API, consulte [CreateCluster](#) na Referência AWS SDK for Java 2.x da API.

Excluir um cluster

O exemplo de código a seguir mostra como excluir um cluster do Amazon Redshift.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Excluir o cluster do .

```
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);
        }
    }
```

- Para obter detalhes da API, consulte [DeleteCluster](#) na Referência AWS SDK for Java 2.x da API.

Descreva seus clusters

O exemplo de código a seguir mostra como descrever seus clusters do Amazon Redshift.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Descreva o 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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeClusters](#) Referência AWS SDK for Java 2.x da API.

Modificar um cluster

O exemplo de código a seguir mostra como modificar um cluster do Amazon Redshift.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Modifique um 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);
    }
}
```

- Para obter detalhes da API, consulte [ModifyCluster](#) Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Rekognition usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Compare faces em uma imagem com uma imagem de referência

O exemplo de código a seguir mostra como comparar faces em uma imagem com uma imagem de referência com o Amazon Rekognition.

Para obter mais informações, consulte [Comparando faces em imagens](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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> uncompered = compareFacesResult.unmatchedFaces();
System.out.println("There was " + uncompered.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);
}
```

```
}
```

- Para obter detalhes da API, consulte [CompareFaces](#) Referência AWS SDK for Java 2.x da API.

Criar uma coleção

O exemplo de código a seguir mostra como criar uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Criação de uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

```
}
```

- Para obter detalhes da API, consulte [CreateCollection](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma coleção

O exemplo de código a seguir mostra como excluir uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Excluindo uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteCollection](#) Referência AWS SDK for Java 2.x da API.

Excluir rostos de uma coleção

O exemplo de código a seguir mostra como excluir faces de uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Excluindo faces de uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteFaces](#) Referência AWS SDK for Java 2.x da API.

Descreva uma coleção

O exemplo de código a seguir mostra como descrever uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Descrição de uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeCollection](#) na Referência AWS SDK for Java 2.x da API.

Detectar faces em uma imagem

O exemplo de código a seguir mostra como detectar rostos em uma imagem com o Amazon Rekognition.

Para obter mais informações, consulte [Detecção de faces em uma imagem](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
}  
}
```

- Para obter detalhes da API, consulte [DetectFaces](#) Referência AWS SDK for Java 2.x da API.

Detectar rótulos em uma imagem

O exemplo de código a seguir mostra como detectar rótulos em uma imagem com o Amazon Rekognition.

Para obter mais informações, consulte [Detecção de rótulos em uma imagem](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DetectLabels](#) a Referência AWS SDK for Java 2.x da API.

Detectar rótulos de moderação em uma imagem

O exemplo de código a seguir mostra como detectar rótulos de moderação em uma imagem com o Amazon Rekognition. Os rótulos de moderação identificam conteúdo que pode ser impróprio para alguns públicos.

Para obter mais informações, consulte [Detecção de imagens inapropriadas](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DetectModerationLabels](#) a Referência AWS SDK for Java 2.x da API.

Detectar texto em uma imagem

O exemplo de código a seguir mostra como detectar texto em uma imagem com o Amazon Rekognition.

Para obter mais informações, consulte [Detecção de texto em uma imagem](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DetectText](#) Referência AWS SDK for Java 2.x da API.

Indexar faces de uma coleção

O exemplo de código a seguir mostra como indexar faces em uma imagem e adicioná-las a uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Adicionar faces a uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [IndexFaces](#) Referência AWS SDK for Java 2.x da API.

Listar coleções

O exemplo de código a seguir mostra como listar as coleções do Amazon Rekognition.

Para obter mais informações, consulte [Listar coleções](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [ListCollections](#) a Referência AWS SDK for Java 2.x da API.

Listar rostos em uma coleção

O exemplo de código a seguir mostra como listar faces em uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Listar faces em uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [ListFaces](#) a Referência AWS SDK for Java 2.x da API.

Reconheça celebridades em uma imagem

O exemplo de código a seguir mostra como reconhecer celebridades em uma imagem com o Amazon Rekognition.

Para obter mais informações, consulte [Reconhecer celebridades em uma imagem](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [RecognizeCelebrities](#) na Referência AWS SDK for Java 2.x da API.

Pesquise rostos em uma coleção

O exemplo de código a seguir mostra como pesquisar faces em uma coleção do Amazon Rekognition que correspondam a outra face da coleção.

Para obter mais informações, consulte [Pesquisando um rosto \(ID facial\)](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [SearchFaces](#) Referência AWS SDK for Java 2.x da API.

Pesquise rostos em uma coleção em comparação com uma imagem de referência

O exemplo de código a seguir mostra como pesquisar rostos em uma coleção do Amazon Rekognition em comparação com uma imagem de referência.

Para obter mais informações, consulte [Procurando por um rosto \(imagem\)](#).

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
}
```

- Para obter detalhes da API, consulte [SearchFacesByImage](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Detecte informações em vídeos

O código de exemplo a seguir mostra como:

- Inicie trabalhos do Amazon Rekognition para detectar elementos como pessoas, objetos e texto em vídeos.
- Verifique o status do trabalho até que o trabalho termine.
- Exiba a lista de elementos detectados por cada tarefa.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Obtenha resultados de celebridades a partir de um vídeo localizado em um bucket do Amazon S3.

```
public static void StartCelebrityDetection(RekognitionClient rekClient,
                                            NotificationChannel channel,
                                            String bucket,
                                            String video){
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3object(s3obj)
```

```
        .build();

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
            .notificationChannel(channel)
            .video(vid0b)
            .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);
}
}
```

Detecte rótulos em um vídeo por meio de uma operação de detecção de rótulos.

```
public static void startLabels(RekognitionClient rekClient,
                               NotificationChannel channel,
                               String bucket,
                               String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidObj)
            .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);
}
}
```

Detecte rostos em um vídeo armazenado em um bucket do Amazon S3.

```
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);
}
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do Amazon S3.

```
public static void startModerationDetection(RekognitionClient rekClient,
                                             NotificationChannel channel,
                                             String bucket,
                                             String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidObj)
            .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);
    }
}
```

Detecte segmentos de sinais técnicos e segmentos de detecção de fotos em um vídeo armazenado em um bucket do Amazon S3.

```
public static void StartSegmentDetection (RekognitionClient rekClient,
                                         NotificationChannel channel,
                                         String bucket,
                                         String video) {

    try {
        S3Object s30bj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3Object(s30bj)
            .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(vid0b)
        .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);
}
}
```

Detecte texto em um vídeo armazenado em um vídeo armazenado em um bucket do Amazon S3.

```
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);
}
}
```

Detecte pessoas em um vídeo armazenado em um vídeo armazenado em um bucket do Amazon S3.

```
public static void startPersonLabels(RekognitionClient rekClient,
                                    NotificationChannel channel,
                                    String bucket,
                                    String video) {
```

```
try {
    S3Object s3Obj = S3Object.builder()
        .bucket(bucket)
        .name(video)
        .build();

    Video vid0b = Video.builder()
        .s3Object(s3Obj)
        .build();

    StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
        .jobTag("DetectingLabels")
        .video(vid0b)
        .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);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.

- [GetCelebrityRecognition](#)
- [GetContentModeration](#)
- [GetLabelDetection](#)
- [GetPersonTracking](#)
- [GetSegmentDetection](#)
- [GetTextDetection](#)
- [StartCelebrityRecognition](#)
- [StartContentModeration](#)
- [StartLabelDetection](#)
- [StartPersonTracking](#)
- [StartSegmentDetection](#)
- [StartTextDetection](#)

Exemplos de registro de domínio do Route 53 usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o registro de domínio AWS SDK for Java 2.x com o Route 53.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Registro de domínios do Hello Route 53

O exemplo de código a seguir mostra como começar a usar o registro de domínios do Route 53.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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 examples performs the following operation:  
 *  
 * 1. Invokes ListPrices for at least one domain type, such as the "com" type and  
 displays the prices for Registration and Renewal.  
 */  
  
public class HelloRoute53 {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) {  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <hostedZoneId> \n\n" +  
            "Where:\n" +  
            "  hostedZoneId - The id value of an existing hosted zone. \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("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);
    }
}
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte [ListPrices](#) Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Verificar disponibilidade de domínios

O exemplo de código a seguir mostra como verificar a disponibilidade de um domínio.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CheckDomainAvailability](#) na Referência AWS SDK for Java 2.x da API.

Verificar a transferibilidade de um domínio

O exemplo de código a seguir mostra como verificar a transferibilidade de um domínio.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [CheckDomainTransferability](#) Referência AWS SDK for Java 2.x da API.

Obter detalhes de um domínio

O exemplo de código a seguir mostra como obter os detalhes de um domínio.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [GetDomainDetail](#) Referência AWS SDK for Java 2.x da API.

Obter detalhes de uma operação

O exemplo de código a seguir mostra como obter detalhes sobre uma operação.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [GetOperationDetail](#) a Referência AWS SDK for Java 2.x da API.

Obter nomes de domínio sugeridos

O exemplo de código a seguir mostra como obter sugestões de nomes de domínio.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainuggestion) {
    try {
        GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
            .domainName(domainuggestion)
            .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);
    }
}
```

- Para obter detalhes da API, consulte [GetDomainSuggestions](#) a Referência AWS SDK for Java 2.x da API.

Listar preços de domínios

O exemplo de código a seguir mostra como listar preços de domínio.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [ListPrices](#) a Referência AWS SDK for Java 2.x da API.

Listar domínios

O exemplo de código a seguir mostra como listar os domínios registrados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListDomains](#) na Referência AWS SDK for Java 2.x da API.

Operações de lista

O exemplo de código a seguir mostra como listar operações.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
```

- Para obter detalhes da API, consulte [ListOperations](#) a Referência AWS SDK for Java 2.x da API.

Registrar um domínio

O exemplo de código a seguir mostra como registrar um domínio.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [RegisterDomain](#) Referência AWS SDK for Java 2.x da API.

Exibir faturamento

O exemplo de código a seguir mostra como visualizar registros de faturamento.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [ViewBilling](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Conceitos básicos de domínios

O código de exemplo a seguir mostra como:

- Liste os domínios atuais e as operações do ano passado.
- Veja o faturamento do ano passado e os preços dos tipos de domínio.
- Receba sugestões de domínio.
- Verifique a disponibilidade e a transferibilidade de um domínio.
- Opcionalmente, solicite o registro de um domínio.
- Obtenha os detalhes de uma operação.
- Opcionalmente, obtenha os detalhes de um domínio.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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.accounts). \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 xxxxxxxx not found in xxxxxxxx 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);
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CheckDomainAvailability](#)
 - [CheckDomainTransferability](#)
 - [GetDomainDetail](#)
 - [GetDomainSuggestions](#)

- [GetOperationDetail](#)
- [ListDomains](#)
- [ListOperations](#)
- [ListPrices](#)
- [RegisterDomain](#)
- [ViewBilling](#)

Exemplos do Amazon S3 usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon S3.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

Ações

Adicionar regras de CORS a um bucket

O exemplo de código a seguir mostra como adicionar regras de compartilhamento de recursos de origem cruzada (CORS) a um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
        }
    }
}
```

- Para obter detalhes da API, consulte [PutBucketCors](#) Referência AWS SDK for Java 2.x da API.

Adicionar uma configuração de ciclo de vida a um bucket

O exemplo de código a seguir mostra como adicionar uma configuração de ciclo de vida a um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [PutBucketLifecycleConfiguration](#) na Referência AWS SDK for Java 2.x da API.

Adicionar uma política a um bucket

O exemplo de código a seguir mostra como adicionar uma política a um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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();
}
```

- Para obter detalhes da API, consulte [PutBucketPolicy](#) Referência AWS SDK for Java 2.x da API.

Copiar um objeto de um bucket para outro

O exemplo de código a seguir mostra como copiar um objeto do S3 de um bucket para outro.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Copie um objeto usando um [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 um [S3 TransferManager](#) para [copiar um objeto](#) de um bucket para outro. Veja o [arquivo completo](#) e [teste](#).

```
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();
}
```

- Para obter detalhes da API, consulte [CopyObject](#) na Referência AWS SDK for Java 2.x da API.

Criar um bucket

O exemplo de código a seguir mostra como criar um bucket do S3.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
}  
}
```

- Para obter detalhes da API, consulte [CreateBucket](#) Referência AWS SDK for Java 2.x da API.

Excluir uma política de um bucket

O exemplo de código a seguir mostra como excluir uma política de um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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);  
    }  
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [DeleteBucketPolicy](#) Referência AWS SDK for Java 2.x da API.

Excluir um bucket vazio

O exemplo de código a seguir mostra como excluir um bucket vazio do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()  
    .bucket(bucket)  
    .build();  
  
s3.deleteBucket(deleteBucketRequest);  
s3.close();
```

- Para obter detalhes da API, consulte [DeleteBucket](#) Referência AWS SDK for Java 2.x da API.

Excluir vários objetos

O exemplo de código a seguir mostra como excluir vários objetos de um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void deleteBucketObjects(S3Client s3, String bucketName) {  
  
    // Upload three sample objects to the specified 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);  
    }  
}
```

```
}
```

- Para obter detalhes da API, consulte [DeleteObjects](#) Referência AWS SDK for Java 2.x da API.

Excluir a configuração de site de um bucket

O exemplo de código a seguir mostra como excluir a configuração do site de um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DeleteBucketWebsite](#) Referência AWS SDK for Java 2.x da API.

Determinar a existência e o tipo de conteúdo de um objeto

O exemplo de código a seguir mostra como determinar a existência e o tipo de conteúdo de um objeto em um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Determine o tipo de conteúdo de um objeto.

```
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);
    }
}
```

Obtenha o status de restauração de um objeto.

```
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);
    }
}
```

- Para obter detalhes da API, consulte [HeadObject](#) Referência AWS SDK for Java 2.x da API.

Fazer download de objetos em um diretório local

O exemplo de código a seguir mostra como fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Use um [S3 TransferManager](#) para [baixar todos os objetos do S3](#) no mesmo bucket do S3. Veja o [arquivo completo e teste](#).

```
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();
}
```

- Para obter detalhes da API, consulte [DownloadDirectory](#) a Referência AWS SDK for Java 2.x da API.

Habilitar notificações

O exemplo de código a seguir mostra como habilitar notificações para um bucket do S3.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void setBucketNotification(S3Client s3Client, String bucketName,
String topicArn, String id) {
```

```
try {
    List<Event> events = new ArrayList<>();
    events.add(Event.S3_OBJECT_CREATED_PUT);

    TopicConfiguration config = TopicConfiguration.builder()
        .topicArn(topicArn)
        .events(events)
        .id(id)
        .build();

    List<TopicConfiguration> topics = new ArrayList<>();
    topics.add(config);

    NotificationConfiguration configuration =
NotificationConfiguration.builder()
        .topicConfigurations(topics)
        .build();

    PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest.builder()
        .bucket(bucketName)
        .notificationConfiguration(configuration)
        .skipDestinationValidation(true)
        .build();

    // Set the bucket notification configuration.
    s3Client.putBucketNotificationConfiguration(configurationRequest);
    System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

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

- Para obter detalhes da API, consulte [PutBucketNotificationConfiguration](#) a Referência AWS SDK for Java 2.x da API.

Obter um objeto de um bucket

O exemplo de código a seguir mostra como ler dados de um objeto em um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Leia dados como uma matriz de bytes usando um [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 um [S3 TransferManager](#) para [baixar um objeto](#) em um bucket do S3 para um arquivo local. Veja o [arquivo completo e teste](#).

```
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 downloadedFilePath) {
        DownloadFileRequest downloadFileRequest =
            DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())
                .destination(Paths.get(downloadedFilePath))
                .build();

        FileDownload downloadFile =
            transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
            downloadFile.completionFuture().join();
        logger.info("Content length [{}]", downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
}
```

Leia etiquetas que pertencem a um objeto usando um [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);
}
}
```

Obtenha um URL para um objeto usando um [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);
    }
}
```

Obtenha um objeto usando o objeto de cliente S3Presigner usando um [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.printStackTrace();
    }
```

```
    }  
}
```

Obtenha um objeto usando um ResponseTransformer objeto e o [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.getObject(objectRequest, ResponseTransformer.toBytes());  
        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);  
    }  
}
```

- Para obter detalhes da API, consulte [GetObject](#) Referência AWS SDK for Java 2.x da API.

Obter a ACL de um bucket

O exemplo de código a seguir mostra como obter a lista de controle de acesso (ACL) de um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [GetBucketAcl](#) Referência AWS SDK for Java 2.x da API.

Obter a política de um bucket

O exemplo de código a seguir mostra como obter a política para um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";  
}
```

- Para obter detalhes da API, consulte [GetBucketPolicy](#) na Referência AWS SDK for Java 2.x da API.

Listar carregamentos fracionados em andamento

O exemplo de código a seguir mostra como listar carregamentos fracionados em andamento em um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListMultipartUploads](#) Referência AWS SDK for Java 2.x da API.

Listar objetos em um bucket

O exemplo de código a seguir mostra como listar objetos em um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;  
}
```

Liste objetos usando paginação.

```
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);
}
}
```

- Para obter detalhes da API, consulte [ListObjectsV2](#) na Referência AWS SDK for Java 2.x da API.

Restaurar uma cópia arquivada de um objeto

O exemplo de código a seguir mostra como restaurar uma cópia arquivada de um objeto em um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [RestoreObject](#) Referência AWS SDK for Java 2.x da API.

Definir uma nova ACL para um bucket

O exemplo de código a seguir mostra como definir uma nova lista de controle de acesso (ACL) para um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [PutBucketAcl](#) Referência AWS SDK for Java 2.x da API.

Definir a configuração de site de um bucket

O exemplo de código a seguir mostra como definir a configuração do site para um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [PutBucketWebsite](#) a Referência AWS SDK for Java 2.x da API.

Carregar um objeto em um bucket

O exemplo de código a seguir mostra como fazer upload de um objeto em um bucket do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Faça upload de um arquivo em um bucket usando um [S3Client](#).

```
// This example uses RequestBody.fromFile to avoid loading the whole file into
memory.
```

```
public static void 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();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket " + bucketName);

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

Use um [S3 TransferManager](#) para fazer [upload de um arquivo](#) em um bucket. Veja o [arquivo completo e teste](#).

```
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();
}
```

Faça upload de um objeto em um bucket e defina etiquetas usando um [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);
}
}
```

Faça upload de um objeto em um bucket e defina metadados usando um [S3Client](#).

```
// This example uses RequestBody.fromFile to avoid loading the whole file into
memory.
public static void 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();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey +" into bucket
"+bucketName);
    }
}
```

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

Faça upload de um objeto em um bucket e defina um valor de retenção de objetos usando um [S3Client](#).

```
public static void setRetentionPeriod(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);
    }
}
```

- Para obter detalhes da API, consulte [PutObject](#) Referência AWS SDK for Java 2.x da API.

Fazer upload em um bucket

O exemplo de código a seguir mostra como fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Use um [S3 TransferManager](#) para [carregar um diretório local](#). Veja o [arquivo completo](#) e [teste](#).

```
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();
}
```

```
}
```

- Para obter detalhes da API, consulte [UploadDirectory](#) Referência AWS SDK for Java 2.x da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar uma URL pré-assinada para o Amazon S3 e fazer o upload de um objeto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * Create a presigned URL for uploading a String object.  
 * @param bucketName - The name of the bucket.  
 * @param keyName - The name of the object.  
 * @return - The presigned URL for an HTTP PUT.  
 */  
public URL createSignedUrlForStringPut(String bucketName, String keyName) {  
    try (S3Presigner presigner = S3Presigner.create()) {  
  
        PutObjectRequest objectRequest = PutObjectRequest.builder()  
            .bucket(bucketName)  
            .key(keyName)  
            .contentType("text/plain")  
            .build();  
  
        PutObjectPresignRequest presignRequest =  
            PutObjectPresignRequest.builder()  
                .signatureDuration(Duration.ofMinutes(10)) // The URL will  
                expire in 10 minutes.  
                .putObjectRequest(objectRequest)
```

```
        .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload to: [{}]", myURL);
        logger.info("Which HTTP method needs to be used when uploading: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url();
    }
}

/***
 * Use the JDK HttpURLConnection (since v1.1) class to upload a String, but you
can
 * use any HTTP client.
 * @param presignedUrl - The presigned URL.
 */
public void useHttpURLConnectionToPutString(URL presignedUrl) {
    try {
        // Create the connection and use it to upload the new object by using
the presigned URL.
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.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();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

/***
 * Use the JDK HttpClient class (since v11) to upload a String,
```

```
* but you can use any HTTP client.  
 * @param presignedUrl - The presigned URL.  
 */  
public void useHttpClientToPutString(URL presignedUrl) {  
    HttpClient httpClient = HttpClient.newHttpClient();  
    try {  
        final HttpResponse<Void> response =  
httpClient.send(HttpRequest.newBuilder()  
            .uri(presignedUrl.toURI())  
            .header("Content-Type", "text/plain")  
            .PUT(HttpRequest.BodyPublishers.ofString("This text was  
uploaded as an object by using a presigned URL."))  
            .build(),  
            HttpResponse.BodyHandlers.discardign());  
        logger.info("HTTP response code is " + response.statusCode());  
    } catch (S3Exception | IOException | URISyntaxException |  
InterruptedException e) {  
        logger.error(e.getMessage(), e);  
    }  
}
```

Conceitos básicos de buckets e objetos

O código de exemplo a seguir mostra como:

- Crie um bucket e faça upload de um arquivo para ele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclua os objetos do bucket e o bucket.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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.
 */
public 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);
}

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.fromFile(new File(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 "";
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Analisar URIs

O exemplo de código a seguir mostra como usar analisar URIs do Amazon S3 para extrair componentes importantes como o nome do bucket e a chave do objeto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Analise um URI do Amazon S3 usando a classe [S3Uri](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client - An S3Client through which you acquire an S3Uri instance.
 * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate S3Uri
 * capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
    logger.info(s3ObjectUrl);
    //Console output: 'https://s3.us-west-1.amazonaws.com/myBucket/resources/
doc.txt?versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri() method.
    URI uri = URI.create(s3ObjectUrl);
    S3Uri s3Uri = s3Utilities.parseUri(uri);

    // If the URI contains no value for the Region, bucket or key, the SDK
    returns an empty Optional.
    // The SDK returns decoded URI values.
```

```
Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77, 88]}'.

// Retrieve the first or all values for a query parameter as shown in the
following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
log("firstMatchingRawQueryParameter", partNumbers);
// Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

/*
Object keys and query parameters with reserved or unsafe characters, must be
URL-encoded.
For example replace whitespace " " with "%20".
```

```
    Valid: "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=%5Bbrackets%5D"
    Invalid: "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"

    Virtual-hosted-style URIs with bucket names that contain a dot, ".", the dot
must not be URL-encoded.
    Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
    Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
*/
}

private static void log(String s3UriElement, Object element) {
    if (element == null) {
        logger.info("{}: {}", s3UriElement, "null");
    } else {
        logger.info("{}: {}", s3UriElement, element.toString());
    }
}
```

Realizar um carregamento fracionado

O exemplo de código a seguir mostra como executar um carregamento fracionado de um objeto do Amazon S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Os exemplos de código usam as importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
```

```
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Use o [Gerenciador de Transferências do S3](#) na parte superior do [cliente do S3 baseado no AWS CRT](#) para realizar de forma transparente um carregamento fracionado quando o tamanho do conteúdo exceder um limite. O limite padrão é 8 MB.

```
public void multipartUploadWithTransferManager(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Use a API [S3Client ou \(API\) S3AsyncClient](#) para realizar um upload de várias partes.

```
public void multipartUploadWithS3Client(String filePath) {
```

```
// Initiate the multipart upload.
CreateMultipartUploadResponse createMultipartUploadResponse =
    s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
String uploadId = createMultipartUploadResponse.uploadId();

// Upload the parts of the file.
int partNumber = 1;
List<CompletedPart> completedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
    long fileSize = file.length();
    int position = 0;
    while (position < fileSize) {
        file.seek(position);
        int read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    logger.error(e.getMessage());
```

```
    }

    // Complete the multipart upload.
    s3Client.completeMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Fazer upload ou download de arquivos grandes

O exemplo de código a seguir mostra como fazer upload ou baixar arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte [Carregar um objeto usando carregamento fracionado](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Chame funções que transferem arquivos de e para um bucket do S3 usando o TransferManager S3.

```
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();
}
```

Carregue um diretório local inteiro.

```
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();
}
```

Carregue um único arquivo.

```
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 de fluxo de tamanho desconhecido

O exemplo de código a seguir mostra como fazer upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Use o [Cliente do S3 baseado em CRT da AWS](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * @param s33CrtAsyncClient - To upload content from a stream of unknown size,
 * use the AWS CRT-based S3 client. For more information, see
 *          https://docs.aws.amazon.com/sdk-for-java/latest/
 * developer-guide/crt-based-s3-client.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 */
```

```
* @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
metadata pertaining to the put object operation.
*/
public PutObjectResponse putObjectFromStream(S3AsyncClient s33CrtAsyncClient,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
    stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

    // AsyncExampleUtils.randomString() returns a random string up to 100
    characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    PutObjectResponse response = responseFuture.join(); // Wait for the
    response.
    logger.info("Object {} uploaded to bucket {}.", key, bucketName);
    return response;
}
}
```

Use o [Gerenciador de transferências do Amazon S3](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;
```

```
import java.io.ByteArrayInputStream;
import java.util.UUID;

/**
 * @param transferManager - To upload content from a stream of unknown size, use
 * the S3TransferManager based on the AWS CRT-based S3 client.
 *          For more information, see https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 * result of the completed upload.
 */
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

    // AsyncExampleUtils.randomString() returns a random string up to 100
    characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
    randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    return upload.completionFuture().join();
}
```

Usar somas de verificação

O exemplo de código a seguir mostra como usar somas de verificação para trabalhar com um objeto do Amazon S3.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Os exemplos de código usam um subconjunto das importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Especifique um algoritmo de soma de verificação para o método `putObject` ao [criar o PutObjectRequest](#).

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test")
    );
}
```

Verifique a soma de verificação do `getObject` método ao [criar o GetObjectRequest](#)

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
    .response();
}
```

Calcule previamente uma soma de verificação para o método `putObject` ao [criar o PutObjectRequest](#).

```
public void putObjectWithPrecalculatedChecksum(String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
```

```
        .checksumSHA256(checksum)),  
        RequestBody.fromFile(Paths.get(filePath)));  
    }  
}
```

Use o [Gerenciador de Transferências do S3](#) na parte superior do [cliente do S3 baseado no AWS CRT](#) para realizar de forma transparente um carregamento fracionado quando o tamanho do conteúdo exceder um limite. O limite padrão é 8 MB.

Você pode especificar um algoritmo de soma de verificação para o SDK usar. Por padrão, o SDK usa o algoritmo CRC32.

```
public void multipartUploadWithChecksumTm(String filePath) {  
    S3TransferManager transferManager = S3TransferManager.create();  
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()  
        .putObjectRequest(b -> b  
            .bucket(bucketName)  
            .key(key)  
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))  
        .source(Paths.get(filePath))  
        .build();  
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);  
    fileUpload.completionFuture().join();  
    transferManager.close();  
}
```

Use a API [S3Client ou \(API\) S3AsyncClient](#) para realizar um upload de várias partes. Se você especificar uma soma de verificação adicional, deverá indicar o algoritmo a ser usado no início do carregamento. Você também deve especificar o algoritmo para a solicitação de cada parte e fornecer a soma de verificação calculada para cada parte após o carregamento.

```
public void multipartUploadWithChecksumS3Client(String filePath) {  
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;  
  
    // Initiate the multipart upload.  
    CreateMultipartUploadResponse createMultipartUploadResponse =  
        s3Client.createMultipartUpload(b -> b  
            .bucket(bucketName)  
            .key(key)  
            .checksumAlgorithm(algorithm)); // Checksum specified on  
    initiation.
```

```
String uploadId = createMultipartUploadResponse.uploadId();

// Upload the parts of the file.
int partNumber = 1;
List<CompletedPart> completedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
    long fileSize = file.length();
    int position = 0;
    while (position < fileSize) {
        file.seek(position);
        int read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .checksumAlgorithm(algorithm) //Checksum specified on each
part.
            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}
```

```
// Complete the multipart upload.  
s3Client.completeMultipartUpload(b -> b  
    .bucket(bucketName)  
    .key(key)  
    .uploadId(uploadId)  
  
.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));  
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo upload de um objeto em um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar o tipo de conteúdo do objeto.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando Java.

```
package example;  
  
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;  
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
```

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

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotification

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

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

Exemplos do S3 Glacier usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o S3 Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Crie um cofre

O exemplo de código a seguir mostra como criar um cofre do Amazon S3 Glacier.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [CreateVault](#) Referência AWS SDK for Java 2.x da API.

Excluir um cofre

O exemplo de código a seguir mostra como excluir um cofre do Amazon S3 Glacier.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteVault](#) Referência AWS SDK for Java 2.x da API.

Excluir um arquivo

O exemplo de código a seguir mostra como excluir um arquivo do Amazon S3 Glacier.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteArchive](#) Referência AWS SDK for Java 2.x da API.

Listar cofres

O exemplo de código a seguir mostra como listar os cofres do Amazon S3 Glacier.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}

}
```

- Para obter detalhes da API, consulte [ListVaults](#) a Referência AWS SDK for Java 2.x da API.

Recupere um inventário do cofre

O exemplo de código a seguir mostra como recuperar um inventário do cofre do Amazon S3 Glacier.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);

}
}
```

- Para obter detalhes da API, consulte [InitiateJob](#) Referência AWS SDK for Java 2.x da API.

Carregar um arquivo em um cofre

O exemplo de código a seguir mostra como fazer upload de um arquivo para um cofre do Amazon S3 Glacier.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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",
        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();
    }
}
```

- Para obter detalhes da API, consulte [UploadArchive](#) a Referência AWS SDK for Java 2.x da API.

SageMaker exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with SageMaker.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá SageMaker

O exemplo de código a seguir mostra como começar a usar o SageMaker.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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 HelloSageMaker {  
    public static void main(String[] args) {  
        Region region = Region.US_WEST_2;  
        SageMakerClient sageMakerClient = SageMakerClient.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
  
        listBooks(sageMakerClient);  
        sageMakerClient.close();  
    }  
  
    public static void listBooks(SageMakerClient sageMakerClient) {  
        try {  
            ListNotebookInstancesResponse notebookInstancesResponse =  
sageMakerClient.listNotebookInstances();  
            List<NotebookInstanceSummary> items =  
notebookInstancesResponse.notebookInstances();  
            for (NotebookInstanceSummary item: items) {  
                System.out.println("The notebook name is:  
"+item.notebookInstanceName());  
            }  
        }  
    }  
}
```

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

- Para obter detalhes da API, consulte [ListNotebookInstances](#) a Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Crie um pipeline

O exemplo de código a seguir mostra como criar ou atualizar um pipeline em SageMaker.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn, String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
```

```
JSONObject jsonObject = (JSONObject) obj;
JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
for (Object stepObj : stepsArray) {
    JSONObject step = (JSONObject) stepObj;
    if (step.containsKey("FunctionArn")) {
        step.put("FunctionArn", functionArn);
    }
}
System.out.println(jsonObject);

// Create the pipeline.
CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
    .pipelineDescription("Java SDK example pipeline")
    .roleArn(roleArn)
    .pipelineName(pipelineName)
    .pipelineDefinition(jsonObject.toString())
    .build();

sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreatePipeline](#)
 - [UpdatePipeline](#)

Apagar um pipeline

O exemplo de código a seguir mostra como excluir um pipeline em SageMaker.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Delete a SageMaker pipeline by name.  
public static void deletePipeline(SageMakerClient sageMakerClient, String  
pipelineName) {  
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()  
        .pipelineName(pipelineName)  
        .build();  
  
    sageMakerClient.deletePipeline(pipelineRequest);  
    System.out.println("**** Successfully deleted "+pipelineName);  
}
```

- Para obter detalhes da API, consulte [DeletePipeline](#) a Referência AWS SDK for Java 2.x da API.

Descreva a execução de um pipeline

O exemplo de código a seguir mostra como descrever a execução de um pipeline em SageMaker.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Check the status of a pipeline execution.  
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,  
String executionArn) throws InterruptedException {  
    String status;
```

```
int index = 0;
do {
    DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
        .pipelineExecutionArn(executionArn)
        .build();

    DescribePipelineExecutionResponse response =
sageMakerClient.describePipelineExecution(pipelineExecutionRequest);
    status = response.pipelineExecutionStatusAsString();
    System.out.println(index +". The Status of the pipeline is "+status);
    TimeUnit.SECONDS.sleep(4);
    index++;
} while ("Executing".equals(status));
System.out.println("Pipeline finished with status "+ status);
}
```

- Para obter detalhes da API, consulte [DescribePipelineExecution](#) a Referência AWS SDK for Java 2.x da API.

Execute um pipeline

O exemplo de código a seguir mostra como iniciar a execução de um pipeline em SageMaker.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl, String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
```

```
.setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
.setPrettyPrinting().create();

// Set up all parameters required to start the pipeline.
List<Parameter> parameters = new ArrayList<>();
Parameter para1 = Parameter.builder()
    .name("parameter_execution_role")
    .value(roleArn)
    .build();

Parameter para2 = Parameter.builder()
    .name("parameter_queue_url")
    .value(queueUrl)
    .build();

String inputJSON = "{\n" +
    "  \"DataSourceConfig\": {\n" +
    "    \"S3Data\": {\n" +
    "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/latlongtest.csv\"\n" +
    "    },\n" +
    "    \"Type\": \"S3_DATA\"\n" +
    "  },\n" +
    "  \"DocumentType\": \"CSV\"\n" +
"}";

System.out.println(inputJSON);

Parameter para3 = Parameter.builder()
    .name("parameter_vej_input_config")
    .value(inputJSON)
    .build();

// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
```

```
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();

System.out.println("parameter_vej_export_config:"+gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
    .build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();

System.out.println("parameter_step_1_vej_config:"+gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}
```

- Para obter detalhes da API, consulte [StartPipelineExecution](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Comece com trabalhos e oleodutos geoespaciais

O código de exemplo a seguir mostra como:

- Configure recursos para um pipeline.
- Configure um pipeline que execute um trabalho geoespacial.
- Inicie a execução de um pipeline.
- Monitore o status da execução.
- Visualize a saída do pipeline.
- Limpe os recursos.

Para obter mais informações, consulte [Criar e executar SageMaker pipelines usando AWS SDKs no Community.aws](#).

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public class SagemakerWorkflow {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    private static String eventSourceMapping = "";  
  
    public static void main(String[] args) throws InterruptedException {  
        final String usage = "\n" +  
            "Usage:\n" +
```

```
        "<sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
<functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
<pipelineName>\n\n" +
    "Where:\n" +
    "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"+
    "    lambdaRoleName - The name of the AWS Lambda role.\n\n"+
    "    functionFileLocation - The file location where the JAR file that
represents the AWS Lambda function is located.\n\n"+
    "    functionName - The name of the AWS Lambda function (for
example, SageMakerExampleFunction).\n\n"+
    "    queueName - The name of the Amazon Simple Queue Service (Amazon
SQS) queue.\n\n"+
    "    bucketName - The name of the Amazon Simple Storage Service (Amazon
S3) bucket.\n\n"+
    "    lnglatData - The file location of the latlongtest.csv file required
for this use case.\n\n"+
    "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"+
    "    pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).\n\n" ;

    if (args.length != 9) {
        System.out.println(usage);
        System.exit(1);
    }

    String sageMakerRoleName = args[0];
    String lambdaRoleName = args[1];
    String functionFileLocation = args[2];
    String functionName = args[3];
    String queueName = args[4];
    String bucketName = args[5];
    String lnglatData = args[6];
    String spatialPipelinePath = args[7];
    String pipelineName = args[8];
    String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

    Region region = Region.US_WEST_2;
    SageMakerClient sageMakerClient = SageMakerClient.builder()
        .region(region)
        .build();

    IamClient iam = IamClient.builder()
        .region(region)
```

```
.build();

LambdaClient lambdaClient = LambdaClient.builder()
    .region(region)
    .build();

SqsClient sqsClient = SqsClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
System.out.println(
    "\nThis example workflow will guide you through setting up and running
an" +
    "\nAmazon SageMaker pipeline. The pipeline uses an AWS Lambda
function and an" +
    "\nAmazon SQS Queue. It runs a vector enrichment reverse geocode job
to" +
    "\nreverse geocode addresses in an input file and store the results
in an export file.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn, handlerName);
String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
System.out.println("The queue URL is "+queueUrl);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Setting up bucket "+bucketName);
if (!checkBucket(s3Client, bucketName)) {
```

```
        setupBucket(s3Client, bucketName);
        System.out.println("Put "+lnglatData +" into "+bucketName);
        putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Now we can create and run our pipeline.");
    setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
    String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn, pipelineName);
    System.out.println("The pipeline execution ARN value is
"+pipelineExecutionARN);
    waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
    System.out.println("Getting output results "+bucketName);
    getOutputResults(s3Client, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The pipeline has completed. To view the pipeline and
runs " +
        "in SageMaker Studio, follow these instructions:" +
        "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Do you want to delete the AWS resources used in this
Workflow? (y/n)");
    Scanner in = new Scanner(System.in);
    String delResources = in.nextLine();
    if (delResources.compareTo("y") == 0) {
        System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
        TimeUnit.SECONDS.sleep(30);
        deleteEventSourceMapping(lambdaClient);
        deleteSQSQueue(sqsClient, queueName);
        listBucketObjects(s3Client, bucketName);
        deleteBucket(s3Client, bucketName);
        deleteLambdaFunction(lambdaClient, functionName);
        deleteLambdaRole(iam, lambdaRoleName);
        deleteSagemakerRole(iam, sageMakerRoleName);
        deletePipeline(sageMakerClient, pipelineName);
    } else {
```

```
        System.out.println("The AWS Resources were not deleted!");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("SageMaker pipeline scenario is complete.");
    System.out.println(DASHES);
}

private static void readObject(S3Client s3Client, String bucketName, String key)
{
    System.out.println("Output file contents: \n");
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
    byte[] byteArray = objectBytes.asByteArray();
    String text = new String(byteArray, StandardCharsets.UTF_8);
    System.out.println("Text output: " + text);
}

// Display some results from the output directory.
public static void getOutputResults(S3Client s3Client, String bucketName) {
    System.out.println("Getting output results {bucketName}.");
    ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
        .bucket(bucketName)
        .prefix("outputfiles/")
        .build();

    ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
    List<S3Object> s3Objects = response.contents();
    for (S3Object object: s3Objects) {
        readObject(s3Client, bucketName, object.key());
    }
}

// Check the status of a pipeline execution.
public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn) throws InterruptedException {
    String status;
    int index = 0;
```

```
do {
    DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
    .pipelineExecutionArn(executionArn)
    .build();

    DescribePipelineExecutionResponse response =
sageMakerClient.describePipelineExecution(pipelineExecutionRequest);
    status = response.pipelineExecutionStatusAsString();
    System.out.println(index +". The Status of the pipeline is "+status);
    TimeUnit.SECONDS.sleep(4);
    index++;
} while ("Executing".equals(status));
System.out.println("Pipeline finished with status "+ status);
}

// Delete a SageMaker pipeline by name.
public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
    DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
        .pipelineName(pipelineName)
        .build();

    sageMakerClient.deletePipeline(pipelineRequest);
    System.out.println("**** Successfully deleted "+pipelineName);
}

// Create a pipeline from the example pipeline JSON.
public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn, String functionArn, String pipelineName) {
    System.out.println("Setting up the pipeline.");
    JSONParser parser = new JSONParser();

    // Read JSON and get pipeline definition.
    try (FileReader reader = new FileReader(filePath)) {
        Object obj = parser.parse(reader);
        JSONObject jsonObject = (JSONObject) obj;
        JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
        for (Object stepObj : stepsArray) {
            JSONObject step = (JSONObject) stepObj;
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArn);
            }
        }
    }
}
```

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

// Create the pipeline.
CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
    .pipelineDescription("Java SDK example pipeline")
    .roleArn(roleArn)
    .pipelineName(pipelineName)
    .pipelineDefinition(jsonObject.toString())
    .build();

sageMakerClient.createPipeline(pipelineRequest);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (IOException | ParseException e) {
    throw new RuntimeException(e);
}
}

// Start a pipeline run with job configurations.
public static String executePipeline(SageMakerClient sageMakerClient, String
bucketName, String queueUrl, String roleArn, String pipelineName) {
    System.out.println("Starting pipeline execution.");
    String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
    String output = "s3://" + bucketName + "/outputfiles/";
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting().create();

    // Set up all parameters required to start the pipeline.
    List<Parameter> parameters = new ArrayList<>();
    Parameter para1 = Parameter.builder()
        .name("parameter_execution_role")
        .value(roleArn)
        .build();

    Parameter para2 = Parameter.builder()
        .name("parameter_queue_url")
        .value(queueUrl)
        .build();

    String inputJSON = "{\n" +
```

```
"  \"DataSourceConfig\": {\n" +
"    \"S3Data\": {\n" +
"      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/latlongtest.csv\"\n" +
+
"    },\n" +
"    \"Type\": \"S3_DATA\"\n" +
"  },\n" +
"  \"DocumentType\": \"CSV\"\n" +
"}";

System.out.println(inputJSON);

Parameter para3 = Parameter.builder()
    .name("parameter_vej_input_config")
    .value(inputJSON)
    .build();

// Create an ExportVectorEnrichmentJobOutputConfig object.
VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
    .s3Uri(output)
    .build();

ExportVectorEnrichmentJobOutputConfig outputConfig =
ExportVectorEnrichmentJobOutputConfig.builder()
    .s3Data(jobS3Data)
    .build();

String gson4 = gson.toJson(outputConfig);
Parameter para4 = Parameter.builder()
    .name("parameter_vej_export_config")
    .value(gson4)
    .build();

System.out.println("parameter_vej_export_config:" + gson.toJson(outputConfig));

// Create a VectorEnrichmentJobConfig object.
ReverseGeocodingConfig reverseGeocodingConfig =
ReverseGeocodingConfig.builder()
    .xAttributeName("Longitude")
    .yAttributeName("Latitude")
    .build();

VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
    .reverseGeocodingConfig(reverseGeocodingConfig)
```

```
.build();

String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":\n{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
Parameter para5 = Parameter.builder()
    .name("parameter_step_1_vej_config")
    .value(para5JSON)
    .build();

System.out.println("parameter_step_1_vej_config:"+gson.toJson(jobConfig));
parameters.add(para1);
parameters.add(para2);
parameters.add(para3);
parameters.add(para4);
parameters.add(para5);

StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
    .pipelineExecutionDescription("Created using Java SDK")
    .pipelineExecutionDisplayName(pipelineName + "-example-execution")
    .pipelineParameters(parameters)
    .pipelineName(pipelineName)
    .build();

StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
return response.pipelineExecutionArn();
}

public static void deleteEventSourceMapping(LambdaClient lambdaClient){
    DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
    .uuid(eventSourceMapping)
    .build();

    lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
}

public static void deleteSagemakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    try {
        for (String policy : sageMakerRolePolicies) {
            // First the policy needs to be detached.
```

```
        DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy)
    .roleName(roleName)
    .build();

    iam.detachRolePolicy(rolePolicyRequest);
}

// 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 deleteLambdaRole(IamClient iam, String roleName) {
    String[] lambdaRolePolicies = getLambdaRolePolicies();
    try {
        for (String policy : lambdaRolePolicies) {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy)
    .roleName(roleName)
    .build();

            iam.detachRolePolicy(rolePolicyRequest);
        }

        // 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);
        }
    }

    // Delete the specific AWS Lambda function.
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("**** "+functionName +" was deleted");

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

// Delete the specific S3 bucket.
public static void deleteBucket(S3Client s3Client, String bucketName) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
    s3Client.deleteBucket(deleteBucketRequest);
    System.out.println("**** "+bucketName +" was deleted.");
}

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());
        }
    }
}
```

```
        deleteBucketObjects(s3, bucketName, myValue.key());
    }

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

}

public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
    toDelete.add(ObjectIdentifier.builder()
        .key(objectName)
        .build());
    try {
        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        s3.deleteObjects(dor);
        System.out.println("*** "+bucketName +" objects were deleted.");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the specific Amazon SQS queue.
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);
        }
    }

    public static void 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("samplefiles/" + objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket "
" + bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void setupBucket(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);
    }
}

// Set up the SQS queue to use with the pipeline.
public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName, String lambdaName) {
    System.out.println("Setting up queue named "+queueName);
    try {
        Map<QueueAttributeName, String> queueAtt = new HashMap<>();
        queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
        queueAtt.put( QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS,
"5");
        queueAtt.put( QueueAttributeName.VISIBILITY_TIMEOUT, "300");
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .attributes(queueAtt)
            .build();

        sqsClient.createQueue(createQueueRequest);
        System.out.println("\nGet queue url");
        GetQueueUrlResponse getQueueUrlResponse =
sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        TimeUnit.SECONDS.sleep(15);

        connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
        System.out.println("Queue ready with Url "+
getQueueUrlResponse.queueUrl());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    return "";
}
```

```
// Connect the queue to the Lambda function as an event source.
public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl, String lambdaName) {
    System.out.println("Connecting the Lambda function and queue for the
pipeline.");
    String queueArn="";

    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);
    GetQueueAttributesRequest attributesRequest=
GetQueueAttributesRequest.builder()
    .queueUrl(queueUrl)
    .attributeNames(atts)
    .build();

    GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAtts = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
        System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
        queueArn = queueAtt.getValue();
    }

    CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
    .eventSourceArn(queueArn)
    .functionName(lambdaName)
    .build();

    CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
    eventSourceMapping = response1.uuid();
    System.out.println("The mapping between the event source and Lambda function
was successful");
}

// Create an AWS Lambda function.
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("SageMaker example function.")
    .code(code)
    .handler(handler)
    .runtime(Runtime.JAVA11)
    .timeout(200)
    .memorySize(1024)
    .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());
return functionResponse.functionArn();

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

public static String createSageMakerRole(IamClient iam, String roleName) {
String[] sageMakerRolePolicies = getSageMakerRolePolicies();
System.out.println("Creating a role to use with SageMaker.");
String assumeRolePolicy = "{" +
"\\"Version\\": \\"2012-10-17\\", " +
"\\"Statement\\": [{" +
"\\"Effect\\": \"Allow\", " +
"\\"Principal\\": {" +
```

```

        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\", " +
        "\"sagemaker-geospatial.amazonaws.com\", " +
        "\"lambda.amazonaws.com\", " +
        "\"s3.amazonaws.com\"" +
    "]" +
"}," +
"\"Action\": \"sts:AssumeRole\"" +
"]]" +
"}";
}

try {
    CreateRoleRequest request = CreateRoleRequest.builder()
        .roleName(roleName)
        .assumeRolePolicyDocument(assumeRolePolicy)
        .description("Created using the AWS SDK for Java")
        .build();

    CreateRoleResponse roleResult = iam.createRole(request);

    // Attach the policies to the role.
    for (String policy : sageMakerRolePolicies) {
        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policy)
        .build();

        iam.attachRolePolicy(attachRequest);
    }

    // Allow time for the role to be ready.
    TimeUnit.SECONDS.sleep(15);
    System.out.println("Role ready with ARN "+roleResult.role().arn());
    return roleResult.role().arn();
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
return "" ;
}

```

```
private static String createLambdaRole(IamClient iam, String roleName) {  
    String [] lambdaRolePolicies = getLambdaRolePolicies();  
    String assumeRolePolicy = "{" +  
        "\"Version\": \"2012-10-17\", " +  
        "\"Statement\": [{" +  
            "\"Effect\": \"Allow\", " +  
            "\"Principal\": {" +  
                "\"Service\": [" +  
                    "\"sagemaker.amazonaws.com\", " +  
                    "\"sagemaker-geospatial.amazonaws.com\", " +  
                    "\"lambda.amazonaws.com\", " +  
                    "\"s3.amazonaws.com\"" +  
                "] " +  
            "}, " +  
            "}], " +  
        "\"Action\": \"sts:AssumeRole\"" +  
    "}] " +  
};  
  
try {  
    CreateRoleRequest request = CreateRoleRequest.builder()  
        .roleName(roleName)  
        .assumeRolePolicyDocument(assumeRolePolicy)  
        .description("Created using the AWS SDK for Java")  
        .build();  
  
    CreateRoleResponse roleResult = iam.createRole(request);  
  
    // Attach the policies to the role.  
    for (String policy : lambdaRolePolicies) {  
        AttachRolePolicyRequest attachRequest =  
AttachRolePolicyRequest.builder()  
            .roleName(roleName)  
            .policyArn(policy)  
            .build();  
  
            iam.attachRolePolicy(attachRequest);  
    }  
  
    // Allow time for the role to be ready.  
    TimeUnit.SECONDS.sleep(15);  
    System.out.println("Role ready with ARN "+roleResult.role().arn());  
    return roleResult.role().arn() ;  
}
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role, String handler) {
    System.out.println("Create an AWS Lambda function used in this workflow.");
    String functionArn;
    try {
        // Does this function already exist.
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
        functionArn = response.configuration().functionArn();

    } catch (LambdaException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
    }
    return functionArn;
}

// Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
method returns true.
public static boolean checkBucket(S3Client s3, String bucketName) {
    try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3.headBucket(headBucketRequest);
        System.out.println(bucketName + " exists");
        return true ;
    } catch (S3Exception e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    return false;
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
new queue
// and returns the ARN value.
public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName, String lambdaName) {
    System.out.println("Creating a queue for this use case.");
    String queueUrl;
    try {
        GetQueueUrlRequest request = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
        queueUrl = response.queueUrl();
        System.out.println(queueUrl);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
    }
    return queueUrl;
}

// Checks to see if the Lambda role exists. If not, this method creates it.
public static String checkLambdaRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS Lambda to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createLambdaRole(iam, roleName);
    }
}
```

```
        }

        return roleArn;
    }

// Checks to see if the SageMaker role exists. If not, this method creates it.
public static String checkSageMakerRole(IamClient iam, String roleName) {
    System.out.println("Creating a role to for AWS SageMaker to use.");
    String roleArn;
    try {
        GetRoleRequest roleRequest = GetRoleRequest.builder()
            .roleName(roleName)
            .build();

        GetRoleResponse response = iam.getRole(roleRequest);
        roleArn = response.role().arn();
        System.out.println(roleArn);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        roleArn = createSageMakerRole(iam, roleName);
    }
    return roleArn;
}

private static String[] getSageMakerRolePolicies() {
    String[] sageMakerRolePolicies = new String[3];
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
"AmazonSageMakerGeospatialFullAccess";
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
    return sageMakerRolePolicies;
}

private static String[] getLambdaRolePolicies() {
    String[] lambdaRolePolicies = new String[5];
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess" ;
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-
role/"+"AmazonSageMakerGeospatialFullAccess";
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-
role/"+"AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-
role/"+"AWSLambdaSQSQueueExecutionRole";
```

```
        return lambdaRolePolicies;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreatePipeline](#)
 - [DeletePipeline](#)
 - [DescribePipelineExecution](#)
 - [StartPipelineExecution](#)
 - [UpdatePipeline](#)

Exemplos do Secrets Manager usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Criar um segredo

O exemplo de código a seguir mostra como criar um segredo do Secrets Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateSecret](#) Referência AWS SDK for Java 2.x da API.

Excluir um segredo

O exemplo de código a seguir mostra como excluir um segredo do Secrets Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteSecreta Referência AWS SDK for Java 2.x da API](#).

Descreva um segredo

O exemplo de código a seguir mostra como descrever um segredo do Secrets Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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( TextStyle.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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeSecret](#) Referência AWS SDK for Java 2.x da API.

Obtenha um valor secreto

O exemplo de código a seguir mostra como obter um valor de segredo do Secrets Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [GetSecretValue](#) a Referência AWS SDK for Java 2.x da API.

Listar segredos

O exemplo de código a seguir mostra como listar segredos do Secrets Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 description is
"+secret.description());
        }
    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [ListSecrets](#) a Referência AWS SDK for Java 2.x da API.

Modifica os detalhes de um segredo

O exemplo de código a seguir mostra como modificar o segredo.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [UpdateSecret](#) Referência AWS SDK for Java 2.x da API.

Coloque um valor em um segredo

O exemplo de código a seguir mostra como colocar um valor em um segredo do Secrets Manager.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void putSecret(SecretsManagerClient secretsClient, String secretName, String secretValue) {  
  
    try {  
        PutSecretValueRequest secretRequest = PutSecretValueRequest.builder()  
            .secretId(secretName)  
            .secretString(secretValue)  
            .build();  
    }
```

```
        secretsClient.putSecretValue(secretRequest);
        System.out.println("A new version was created.");

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

- Para obter detalhes da API, consulte [PutSecretValue](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon SES usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Listar modelos de e-mail

O exemplo de código a seguir mostra como lista modelos existentes de e-mail do Amazon SES.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListTemplates](#) Referência AWS SDK for Java 2.x da API.

Listar identidades

O exemplo de código a seguir mostra como listar as identidades do Amazon SES.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void listSESIentities(SesClient client) {  
  
    try {  
        ListIdentitiesResponse identitiesResponse = client.listIdentities();  
        List<String> identities = identitiesResponse.getIdentities();  
        for (String identity: identities) {  
            System.out.println("The identity is "+identity);  
        }  
  
    } catch (SesException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [ListIdentities](#) Referência AWS SDK for Java 2.x da API.

Enviar e-mail

O exemplo de código a seguir mostra como enviar e-mails com o Amazon SES.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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");
}
```

- Para obter detalhes da API, consulte [SendEmail](#) na Referência AWS SDK for Java 2.x da API.

Enviar e-mails com base em modelo

O exemplo de código a seguir mostra como enviar e-mails com base em modelo com o Amazon SES.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [SendTemplatedEmail](#) a Referência AWS SDK for Java 2.x da API.

Exemplos da API v2 do Amazon SES usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK for Java 2.x API v2 do Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Enviar um e-mail

O exemplo de código a seguir mostra como enviar um e-mail com a API v2 do Amazon SES.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Envia uma mensagem.

```
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);
}
}
```

- Para obter detalhes da API, consulte [SendEmail](#) na Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon SNS usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Hello Amazon SNS

Os exemplos de código a seguir mostram como começar a usar o Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
package com.example.sns;

import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
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.paginators.ListTopicsIterable;

public class HelloSNS {

    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
```

```
        .flatMap(r -> r.topics().stream())
        .forEach(content -> System.out.println(" Topic ARN: " +
content.topicArn()));

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

- Para obter detalhes da API, consulte [ListTopics](#) Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Adicionar tags a um tópico

O exemplo de código a seguir mostra como adicionar tags a um tópico do Amazon SNS.

SDK for Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [TagResource](#) Referência AWS SDK for Java 2.x da API.

Verificar se um número de telefone saiu

O exemplo de código a seguir mostra como verificar se um número de telefone optou por não receber mensagens do Amazon SNS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [CheckIfPhoneNumberIsOptedOut](#) Referência AWS SDK for Java 2.x da API.

Confirmar que um proprietário de endpoint deseja receber mensagens

O exemplo de código a seguir mostra como confirmar que o proprietário de um endpoint deseja receber mensagens do Amazon SNS validando o token enviado ao endpoint por uma ação anterior de inscrição.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [ConfirmSubscription](#) Referência AWS SDK for Java 2.x da API.

Criar um tópico

O exemplo de código a seguir mostra como criar um tópico do Amazon SNS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateTopic](#) Referência AWS SDK for Java 2.x da API.

Excluir uma assinatura

O exemplo de código a seguir mostra como excluir uma assinatura do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [Unsubscribe](#) na Referência da API do AWS SDK for Java 2.x.

Excluir um tópico

O exemplo de código a seguir mostra como excluir um tópico do Amazon SNS e todas as assinaturas desse tópico.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteTopic](#) a Referência AWS SDK for Java 2.x da API.

Obter as propriedades de um tópico

O exemplo de código a seguir mostra como obter as propriedades de um tópico do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [GetTopicAttributes](#) a Referência AWS SDK for Java 2.x da API.

Obter as configurações para enviar mensagens SMS

O exemplo de código a seguir mostra como obter as configurações para enviar mensagens SMS do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void getSNSAttributes(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");  
}
```

- Para obter detalhes da API, consulte [GetSMSAttributes](#) na Referência de API do AWS SDK for Java 2.x.

Listar números de telefone que saíram

O exemplo de código a seguir mostra como listar números de telefone que optaram por não receber mensagens do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void list0pts( 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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListPhoneNumbersOptedOut](#) na Referência AWS SDK for Java 2.x da API.

Listar os assinantes de um tópico

O exemplo de código a seguir mostra como recuperar a lista de assinantes de um tópico do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListSubscriptions](#) a Referência AWS SDK for Java 2.x da API.

Listar tópicos

O exemplo de código a seguir mostra como listar tópicos do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [ListTopics](#) Referência AWS SDK for Java 2.x da API.

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [Publish](#) na Referência da API do AWS SDK for Java 2.x.

Publicar em um tópico

O exemplo de código a seguir mostra como publicar mensagens em um tópico do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

```
}
```

- Para obter detalhes da API, consulte [Publish](#) na Referência da API do AWS SDK for Java 2.x.

Definir uma política de filtro

O exemplo de código a seguir mostra como definir uma política de filtro do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    } catch (AmazonSNSException e) {  
        System.out.println(e.getMessage());  
    }  
}
```

```
// 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);  
}  
}
```

- Para obter detalhes da API, consulte [SetSubscriptionAttributes](#) a Referência AWS SDK for Java 2.x da API.

Definir as configurações padrão para enviar mensagens SMS

O exemplo de código a seguir mostra como definir as configurações padrão para enviar mensagens SMS usando o Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
        }
    }
}
```

- Para receber detalhes da API, consulte [SetSMSAttributes](#) na Referência da API do AWS SDK for Java 2.x.

Definir atributos de tópicos

O exemplo de código a seguir mostra como definir atributos de tópicos do Amazon SNS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [SetTopicAttributes](#) Referência AWS SDK for Java 2.x da API.

Inscrever uma função do Lambda em um tópico

O exemplo de código a seguir mostra como assinar uma função Lambda para que ela receba notificações de um tópico do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [Subscribe](#) na Referência da API do AWS SDK for Java 2.x.

Inscrever um endpoint HTTP em um tópico

O exemplo de código a seguir mostra como assinar um endpoint HTTP ou HTTPS para que ele receba notificações de um tópico do Amazon SNS.

SDK para Java 2.x



Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [Subscribe](#) na Referência da API do AWS SDK for Java 2.x.

Inscrever um endereço de e-mail em um tópico

O exemplo de código a seguir mostra como inscrever um endereço de e-mail em um tópico do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [Subscribe](#) na Referência da API do AWS SDK for Java 2.x.

Cenários

Criar um endpoint de plataforma para notificações por push

O exemplo de código a seguir mostra como criar um endpoint de plataforma para notificações por push do Amazon SNS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
}
```

Criar e publicar em um tópico FIFO

Os exemplos de código a seguir mostram como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O [teste](#) verifica o recebimento da mensagem em cada fila. O [exemplo completo](#) também mostra a adição de políticas de acesso e exclui os recursos no final.

```
public class PriceUpdateExample {  
    public final static SnsClient snsClient = SnsClient.create();  
    public final static SqsClient sqsClient = SqsClient.create();  
  
    public static void main(String[] args) {  
  
        final String usage = "\n" +  
            "Usage: " +  
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>  
<analyticsQueueName>\n\n" +  
            "Where:\n" +  
            "    fifoTopicName - The name of the FIFO topic that you want to  
create. \n\n" +  
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be  
created for the wholesale consumer. \n\n" +  
            "    retailQueueARN - The name of a SQS FIFO queue that will created  
for the retail consumer. \n\n" +  
            "    analyticsQueueARN - The name of a SQS standard queue that will  
be created for the analytics consumer. \n\n";  
        if (args.length != 4) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        final String fifoTopicName = args[0];  
        final String wholeSaleQueueName = args[1];  
        final String retailQueueName = args[2];
```

```
final String analyticsQueueName = args[3];

// For convenience, the QueueData class holds metadata about a queue: ARN,
URL, name and type.
List<QueueData> queues = List.of(
    new QueueData(wholeSaleQueueName, QueueType.FIFO),
    new QueueData(retailQueueName, QueueType.FIFO),
    new QueueData(analyticsQueueName, QueueType.Standard));

// Create queues.
createQueues(queues);

// Create a topic.
String topicARN = createFIFOTopic(fifoTopicName);

// Subscribe each queue to the topic.
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "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);

        return topicArn;

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

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqS")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO topic.
        SubscribeResponse subscribeResponse =
        snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        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)
        .messageDuplicationId(dedupId)
        .messageAttributes(attributes)
        .build();

    final PublishResponse response = snsClient.publish(pubRequest);
    System.out.println(response.messageId());
    System.out.println(response.sequenceNumber());
    System.out.println("Message was published to " + topicArn);

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateTopic](#)
 - [Publicar](#)
 - [Assinar](#)

Publicar mensagens SMS em um tópico

O exemplo de código a seguir mostra como:

- Crie um tópico do Amazon SNS.
- Inscrever números de telefone no tópico.
- Publicar mensagens SMS no tópico para que todos os números de telefone inscritos recebam a mensagem de uma só vez.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Criar um tópico e retorne seu 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 "";  
}
```

Inscreva um endpoint em um tópico.

```
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);
    }
}
```

Defina atributos na mensagem, como o ID do remetente, o preço máximo e seu tipo. Os atributos de mensagem são opcionais.

```
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();
        setSMSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSMSAttributes( 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);
    }
}
```

Publique uma mensagem em um tópico. A mensagem é enviada para todos os assinantes.

```
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);
    }
}
```

Exemplos do Amazon SQS usando SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá, Amazon SQS

Os exemplos de código a seguir mostram como começar a usar o Amazon SQS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * 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 HelloSQS {

    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
```

```
try {
    ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
    listQueues.stream()
        .flatMap(r -> r.queueUrls().stream())
        .forEach(content -> System.out.println(" Queue URL: " +
content.toLowerCase()));
}

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

- Para obter detalhes da API, consulte [ListQueues](#) Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

Ações

Criar uma fila

O exemplo de código a seguir mostra como criar uma fila do Amazon SQS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String createQueue(SqsClient sqsClient, String queueName ) {

    try {
```

```
System.out.println("\nCreate Queue");

CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
    .queueName(queueName)
    .build();

sqscClient.createQueue(createQueueRequest);

System.out.println("\nGet queue url");

GetQueueUrlResponse getQueueUrlResponse =
sqscClient.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(SqscClient sqscClient) {

System.out.println("\nList Queues");
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqscClient.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(SqscClient sqscClient, 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);
    }
}
```

- Para obter detalhes da API, consulte [CreateQueuea Referência AWS SDK for Java 2.x da API.](#)

Excluir uma mensagem de uma fila

O exemplo de código a seguir mostra como excluir uma mensagem de uma fila do Amazon SQS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
    .queueUrl(queueUrl)
```

```
        .receiptHandle(message.receiptHandle())
        .build();
    sqsClient.deleteMessage(deleteMessageRequest);
}
```

- Para obter detalhes da API, consulte [DeleteMessage](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma fila

O exemplo de código a seguir mostra como excluir uma fila do Amazon SQS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DeleteQueue](#) Referência AWS SDK for Java 2.x da API.

Obter o URL de uma fila

O exemplo de código a seguir mostra como obter a URL de uma fila do Amazon SQS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) Referência AWS SDK for Java 2.x da API.

Listar filas

O exemplo de código a seguir mostra como listar filas do Amazon SQS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
```

- Para obter detalhes da API, consulte [ListQueues](#) Referência AWS SDK for Java 2.x da API.

Receber mensagens de uma fila

O exemplo de código a seguir mostra como receber mensagens de uma fila do Amazon SQS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}
```

- Para obter detalhes da API, consulte [ReceiveMessage](#) Referência AWS SDK for Java 2.x da API.

Enviar um lote de mensagens para uma fila

O exemplo de código a seguir mostra como enviar um lote de mensagens para uma fila do Amazon SQS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
```

- Para obter detalhes da API, consulte [Send Message Batch](#) Referência AWS SDK for Java 2.x da API.

Enviar uma mensagem para uma fila

O exemplo de código a seguir mostra como enviar uma mensagem para uma fila do Amazon SQS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [SendMessage](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Criar e publicar em um tópico FIFO

Os exemplos de código a seguir mostram como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O [teste](#) verifica o recebimento da mensagem em cada fila. O [exemplo completo](#) também mostra a adição de políticas de acesso e exclui os recursos no final.

```
public class PriceUpdateExample {  
    public final static SnsClient snsClient = SnsClient.create();  
    public final static SqsClient sqsClient = SqsClient.create();  
  
    public static void main(String[] args) {  
  
        final String usage = "\n" +  
            "Usage: " +  
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>  
<analyticsQueueName>\n\n" +  
            "Where:\n" +  
            "    fifoTopicName - The name of the FIFO topic that you want to  
create. \n\n" +  
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be  
created for the wholesale consumer. \n\n" +  
            "    retailQueueARN - The name of a SQS FIFO queue that will created  
for the retail consumer. \n\n" +  
            "    analyticsQueueARN - The name of a SQS standard queue that will  
be created for the analytics consumer. \n\n";  
        if (args.length != 4) {  
            System.out.println(usage);  
        }  
    }  
}
```

```
        System.exit(1);
    }

    final String fifoTopicName = args[0];
    final String wholeSaleQueueName = args[1];
    final String retailQueueName = args[2];
    final String analyticsQueueName = args[3];

    // For convenience, the QueueData class holds metadata about a queue: ARN,
    URL, name and type.
    List<QueueData> queues = List.of(
        new QueueData(wholeSaleQueueName, QueueType.FIFO),
        new QueueData(retailQueueName, QueueType.FIFO),
        new QueueData(analyticsQueueName, QueueType.Standard)));

    // Create queues.
    createQueues(queues);

    // Create a topic.
    String topicARN = createFIFOTopic(fifoTopicName);

    // Subscribe each queue to the topic.
    subscribeQueues(queues, topicARN);

    // Allow the newly created topic to send messages to the queues.
    addAccessPolicyToQueuesFINAL(queues, topicARN);

    // Publish a sample price update message with payload.
    publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
    "Consumables");

    // Clean up resources.
    deleteSubscriptions(queues);
    deleteQueues(queues);
    deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "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);

        return topicArn;

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

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqS")
            .build();

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
        FIFO topic.
        SubscribeResponse subscribeResponse =
        snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
```

```
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)
    .messageDuplicationId(dedupId)
    .messageAttributes(attributes)
    .build();

final PublishResponse response = snsClient.publish(pubRequest);
System.out.println(response.messageId());
System.out.println(response.sequenceNumber());
System.out.println("Message was published to " + topicArn);

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateTopic](#)
 - [Publicar](#)
 - [Assinar](#)

Exemplos sem servidor

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções Lambda que recebem eventos de uma fila SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de itens em lote do SQS com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
    }
}
```

```
        }
    }
    return new SQSBatchResponse(batchItemFailures);
}
}
```

Exemplos de Step Functions usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Step Functions.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Hello Step Functions

Os exemplos de código a seguir mostram como começar a usar o Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Versão Java do 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);
    }
}
```

- Para obter detalhes da API, consulte [ListStateMachines](#) Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar uma máquina de estado

O exemplo de código a seguir mostra como criar uma máquina de estado Step Functions.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateStateMachine](#) Referência AWS SDK for Java 2.x da API.

Crie uma atividade

O exemplo de código a seguir mostra como criar uma atividade Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [CreateActivity](#) na Referência AWS SDK for Java 2.x da API.

Excluir uma máquina de estado

O exemplo de código a seguir mostra como excluir uma máquina de estado do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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.");
}
```

- Para obter detalhes da API, consulte [DeleteStateMachine](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma atividade

O exemplo de código a seguir mostra como excluir uma atividade de Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

```
    }  
}
```

- Para obter detalhes da API, consulte [DeleteActivity](#) Referência AWS SDK for Java 2.x da API.

Descreva uma máquina de estado

O exemplo de código a seguir mostra como descrever uma máquina de estado Step Functions.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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());  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeStateMachine](#) na Referência AWS SDK for Java 2.x da API.

Descreva a execução de uma máquina de estado

O exemplo de código a seguir mostra como descrever a execução de uma máquina de estado do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeExecution](#) na Referência AWS SDK for Java 2.x da API.

Obter dados de tarefas para uma atividade

O exemplo de código a seguir mostra como obter dados de tarefas para uma atividade de Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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;
}

/// <summary>
```

```
/// Stop execution of a Step Functions workflow.  
/// </summary>  
/// <param name="executionArn">The Amazon Resource Name (ARN) of  
/// the Step Functions execution to stop.</param>  
/// <returns>A Boolean value indicating the success of the action.</returns>  
public async Task<bool> StopExecution(string executionArn)  
{  
    var response =  
        await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest  
<{ ExecutionArn = executionArn }>);  
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
}
```

- Para obter detalhes da API, consulte [GetActivityTask](#) Referência AWS SDK for Java 2.x da API.

Listar atividades

O exemplo de código a seguir mostra como listar as atividades do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [ListActivities](#) Referência AWS SDK for Java 2.x da API.

Listar estados executados pela máquina

O exemplo de código a seguir mostra como listar as execuções de estado da máquina de Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
        }
    }
```

- Para obter detalhes da API, consulte [ListExecutions](#) a Referência AWS SDK for Java 2.x da API.

Listar máquinas de estado

O exemplo de código a seguir mostra como listar as máquinas de estado do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [ListStateMachines](#) Referência AWS SDK for Java 2.x da API.

Enviar uma resposta bem-sucedida para uma tarefa

O exemplo de código a seguir mostra como enviar uma resposta bem-sucedida para uma tarefa do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [SendTaskSuccess](#) Referência AWS SDK for Java 2.x da API.

Inicie a execução de uma máquina de estado

O exemplo de código a seguir mostra como iniciar a execução de uma máquina de estado do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [StartExecution](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Comece a usar máquinas de estado

O código de exemplo a seguir mostra como:

- Crie uma atividade.

- Crie uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Execute a máquina de estado e responda à atividade com a entrada do usuário.
- Obtenha o status final e a saída após a conclusão da execução e, em seguida, limpe os recursos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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.js")
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 "";
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.
 - [CreateActivity](#)
 - [CreateStateMachine](#)

- [DeleteActivity](#)
- [DeleteStateMachine](#)
- [DescribeExecution](#)
- [DescribeStateMachine](#)
- [GetActivityTask](#)
- [ListActivities](#)
- [ListStateMachines](#)
- [SendTaskSuccess](#)
- [StartExecution](#)
- [StopExecution](#)

AWS STS exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Assumir uma função

O exemplo de código a seguir mostra como assumir uma função com AWS STS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [AssumeRole](#) a Referência AWS SDK for Java 2.x da API.

AWS Support exemplos usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS Support.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá, AWS Support

O exemplo de código a seguir mostra como começar a usar o AWS Support.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
/**  
 * 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());  
                }  
            }  
        } catch (Exception e) {  
            System.out.println("An error occurred: " + e.getMessage());  
        }  
    }  
}
```

```
        }
        index++ ;
    }

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

}
```

- Para obter detalhes da API, consulte [DescribeServices](#) a Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Adicionar uma comunicação a um caso

O exemplo de código a seguir mostra como adicionar uma AWS Support comunicação com um anexo a um caso de suporte.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [AddCommunicationToCase](#) a Referência AWS SDK for Java 2.x da API.

Adicionar um anexo a um conjunto

O exemplo de código a seguir mostra como adicionar um AWS Support anexo a um conjunto de anexos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [AddAttachmentsToSet](#) Referência AWS SDK for Java 2.x da API.

Criar um caso

O exemplo de código a seguir mostra como criar um novo AWS Support caso.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";  
}
```

- Para obter detalhes da API, consulte [CreateCase](#) a Referência AWS SDK for Java 2.x da API.

Descrever um anexo

O exemplo de código a seguir mostra como descrever um anexo para um AWS Support caso.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DescribeAttachment](#) na Referência AWS SDK for Java 2.x da API.

Descrever casos

O exemplo de código a seguir mostra como descrever AWS Support casos.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
}
}
```

- Para obter detalhes da API, consulte [DescribeCases](#) Referência AWS SDK for Java 2.x da API.

Descrever as comunicações

O exemplo de código a seguir mostra como descrever AWS Support as comunicações de um caso.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [DescribeCommunications](#) a Referência AWS SDK for Java 2.x da API.

Descrever os serviços

O exemplo de código a seguir mostra como descrever a lista de AWS serviços.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
// 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;  
}
```

- Para obter detalhes da API, consulte [DescribeServices](#) a Referência AWS SDK for Java 2.x da API.

Descrever os níveis de gravidade

O exemplo de código a seguir mostra como descrever os níveis de AWS Support severidade.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [DescribeSeverityLevels](#) a Referência AWS SDK for Java 2.x da API.

Resolver caso

O exemplo de código a seguir mostra como resolver um AWS Support caso.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [ResolveCasea](#) Referência AWS SDK for Java 2.x da API.

Cenários

Conceitos básicos de casos

O código de exemplo a seguir mostra como:

- Obtenha e exiba os serviços disponíveis e os níveis de gravidade dos casos.
- Crie um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obtenha e exiba uma lista de casos em aberto para o dia atual.
- Adicione um conjunto de anexos e uma comunicação ao novo caso.
- Descreva o novo anexo e a comunicação para o caso.
- Resolva o caso.
- Obtenha e exiba uma lista de casos resolvidos para o dia atual.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute várias operações do AWS Support.

```
/**
 * 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;
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK for Java 2.x.

- [AddAttachmentsToSet](#)
- [AddCommunicationToCase](#)
- [CreateCase](#)
- [DescribeAttachment](#)
- [DescribeCases](#)
- [DescribeCommunications](#)
- [DescribeServices](#)
- [DescribeSeverityLevels](#)
- [ResolveCase](#)

Exemplos do Systems Manager usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Systems Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Adicionar um parâmetro

O exemplo de código a seguir mostra como adicionar um parâmetro do Systems Manager.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void putParaValue(SsmClient ssmClient, String paraName, String value) {
    try {
        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(paraName)
            .type(ParameterType.STRING)
            .value(value)
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.println("The parameter was successfully added.");
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [PutParameter](#) na Referência AWS SDK for Java 2.x da API.

Crie um novo OpsItem

O exemplo de código a seguir mostra como criar um novo OpsItem.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static String createNewOpsItem( SsmClient ssmClient,
                                         String title,
                                         String source,
                                         String category,
                                         String severity) {

    try {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
            .description("Created by the SSM Java API")
            .title(title)
            .source(source)
            .category(category)
            .severity(severity)
            .build();

        CreateOpsItemResponse itemResponse =
ssmClient.createOpsItem(opsItemRequest);
        return itemResponse.opsItemId();

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

- Para obter detalhes da API, consulte [CreateOpsItem](#) Referência AWS SDK for Java 2.x da API.

Descreva um OpsItem

O exemplo de código a seguir mostra como descrever um OpsItem.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void describeItems(SsmClient ssmClient) {  
  
    try {  
        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()  
            .maxResults(10)  
            .build();  
  
        DescribeOpsItemsResponse itemsResponse =  
ssmClient.describeOpsItems(itemsRequest);  
        List<OpsItemSummary> items = itemsResponse.opsItemSummaries();  
        for (OpsItemSummary item: items) {  
            System.out.println("The item title is "+item.title());  
        }  
  
    } catch (SsmException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeOpsItems](#) na Referência AWS SDK for Java 2.x da API.

Obtenha informações sobre parâmetros

O exemplo de código a seguir mostra como obter informações sobre os parâmetros do Systems Manager.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void getParaValue(SsmClient ssmClient, String paraName) {
```

```
try {
    GetParameterRequest parameterRequest = GetParameterRequest.builder()
        .name(paraName)
        .build();

    GetParameterResponse parameterResponse =
    ssmClient.getParameter(parameterRequest);
    System.out.println("The parameter value is
"+parameterResponse.parameter().value());

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

- Para obter detalhes da API, consulte [DescribeParameters](#) Referência AWS SDK for Java 2.x da API.

Atualiza e OpsItem

O exemplo de código a seguir mostra como atualizar um OpsItem.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
public static void setOpsItemStatus(SsmClient ssmClient, String opsID) {

    try {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
            .opsItemId(opsID)
            .status(OpsItemStatus.RESOLVED)
            .build();
    }
}
```

```
        ssmClient.updateOpsItem(opsItemRequest);

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

- Para obter detalhes da API, consulte [UpdateOpsItem](#) Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Textract usando o SDK for Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Textract.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, você pode ver as ações contextualizadas em seus devidos cenários e exemplos de serviços cruzados.

Cenários são exemplos de código que mostram como realizar uma tarefa específica chamando várias funções dentro do mesmo serviço.

Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

Ações

Analisar um documento

O exemplo de código a seguir mostra como analisar um documento usando o Amazon Textract.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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);
    }
}
```

- Para obter detalhes da API, consulte [AnalyzeDocument](#) Referência AWS SDK for Java 2.x da API.

Detectar texto em um documento

O exemplo de código a seguir mostra como detectar texto em um documento usando o Amazon Textract.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Detecte texto de um documento de entrada.

```
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);
    }
}
```

Detecte texto de um documento localizado em um bucket do Amazon S3.

```
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);
    }
}
```

- Para obter detalhes da API, consulte [DetectDocumentTexta](#) Referência AWS SDK for Java 2.x da API.

Iniciar a análise assíncrona de um documento

O exemplo de código a seguir mostra como iniciar a análise assíncrona de um documento usando o Amazon Textract.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [StartDocumentAnalysis](#) Referência AWS SDK for Java 2.x da API.

Exemplos de serviços cruzados usando o SDK for Java 2.x

Os aplicativos de exemplo a seguir usam o AWS SDK for Java 2.x para trabalhar em vários Serviços da AWS.

Os exemplos de serviços cruzados visam um nível avançado de experiência para ajudar você a começar a criar aplicativos.

Exemplos

- [Criar uma aplicação para enviar dados para uma tabela do DynamoDB](#)
- [Crie um chatbot Amazon Lex para engajar os visitantes do seu site](#)
- [Criar uma aplicação de publicação e assinatura que traduz mensagens](#)
- [Crie um aplicativo web que envie e recupere mensagens usando o Amazon SQS](#)

- [Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos](#)
- [Criar uma aplicação Web para monitorar dados do DynamoDB](#)
- [Criar um rastreador de itens do Amazon Redshift](#)
- [Crie um rastreador de itens de trabalho do Aurora Sem Servidor](#)
- [Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio](#)
- [Detectar EPI em imagens com o Amazon Rekognition usando um AWS SDK](#)
- [Detectar objetos em imagens com o Amazon Rekognition usando um AWS SDK](#)
- [Detectar pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK](#)
- [Publicação de mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK](#)
- [Usar o API Gateway para invocar uma função do Lambda](#)
- [Usar Step Functions para invocar funções do Lambda](#)
- [Usar eventos programados para invocar uma função do Lambda](#)

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

SDK para Java 2.x

Mostra como criar uma aplicação Web dinâmica que envia dados usando a API Java do Amazon DynamoDB e envia uma mensagem de texto usando a API Java do Amazon Simple Notification Service.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Crie um chatbot Amazon Lex para engajar os visitantes do seu site

SDK para Java 2.x

Mostra como usar a API do Amazon Lex para criar um Chatbot em um aplicativo da web para engajar os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Criar uma aplicação de publicação e assinatura que traduz mensagens

SDK para Java 2.x

Mostra como usar a API Java do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo que usa a API Java Async, consulte o exemplo completo em [GitHub](#)

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Crie um aplicativo web que envie e recupere mensagens usando o Amazon SQS

SDK para Java 2.x

Mostra como usar a API do Amazon SQS para desenvolver uma API Spring REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação Web para monitorar dados do DynamoDB

SDK para Java 2.x

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB

- Amazon SES

Criar um rastreador de itens do Amazon Redshift

SDK para Java 2.x

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. [GitHub](#)

Serviços usados neste exemplo

- Amazon Redshift
- Amazon SES

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. [GitHub](#)

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. [GitHub](#)

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês usando o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em [GitHub](#).

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Detectar EPI em imagens com o Amazon Rekognition usando um AWS SDK

SDK para Java 2.x

Mostra como criar uma função do AWS Lambda que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar objetos em imagens com o Amazon Rekognition usando um AWS SDK

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Publicação de mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK

SDK para Java 2.x

Demonstra mensagens com tópicos e filas usando o Amazon Simple Notification Service (Amazon SNS) e o Amazon Simple Queue Service (Amazon SQS).

Para obter o código-fonte completo e as instruções que demonstram mensagens com tópicos e filas no Amazon SNS e no Amazon SQS, veja o exemplo completo em. [GitHub](#)

Serviços usados neste exemplo

- Amazon SNS
- Amazon SQS

Usar o API Gateway para invocar uma função do Lambda

SDK para Java 2.x

Mostra como criar uma função do AWS Lambda usando a API de runtime de Java do Lambda.

Este exemplo invoca diferentes serviços da AWS para lidar com um caso de uso específico. Este exemplo demonstra como criar uma função do Lambda chamada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon SNS

Usar Step Functions para invocar funções do Lambda

SDK para Java 2.x

Mostra como criar um AWS fluxo de trabalho sem servidor usando o AWS Step Functions e o AWS SDK for Java 2.x. Cada etapa do fluxo de trabalho é implementada usando uma função do AWS Lambda.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Usar eventos programados para invocar uma função do Lambda

SDK para Java 2.x

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca diferentes serviços da AWS para lidar com um caso de uso específico. Este exemplo demonstra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em [GitHub](#).

Serviços usados neste exemplo

- DynamoDB

- EventBridge
- Lambda
- Amazon SNS

Segurança para o AWS SDK for Java

A segurança da nuvem na Amazon Web Services (AWS) é a nossa maior prioridade. Como cliente da AWS, você se contará com um datacenter e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança. A segurança é uma responsabilidade compartilhada entre a AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como a Segurança da nuvem e a Segurança na nuvem.

Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa todos os serviços oferecidos na Nuvem AWS e por fornecer serviços que você pode usar com segurança. Nossa responsabilidade de segurança é a maior prioridade na AWS, e a eficácia da nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos [Programas de Compatibilidade da AWS](#).

Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você estiver usando e por outros fatores, incluindo a confidencialidade dos dados, os requisitos da organização e as leis e regulamentos aplicáveis.

Esse produto ou serviço da AWS segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) compatíveis. Para obter informações de segurança sobre o serviço da AWS, consulte a [página de documentação de segurança do serviço da AWS](#) e [Serviços da AWS que estão no escopo dos esforços de conformidade da AWS por programa de conformidade](#).

Tópicos

- [Proteção de dados em AWS SDK for Java 2.x](#)
- [Trabalhando com TLS no SDK for Java](#)
- [Gerenciamento de identidade e acesso](#)
- [Validação de conformidade para este produto ou serviço da AWS](#)
- [Resiliência para este produto ou serviço da AWS](#)
- [Segurança da infraestrutura para esse produto ou serviço da AWS](#)

Proteção de dados em AWS SDK for Java 2.x

O [modelo de responsabilidade compartilhada](#) da AWS se aplica à proteção de dados no AWS SDK for Java. Conforme descrito nesse modelo, a AWS é responsável por proteger a infraestrutura global

que executa toda a Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Esse conteúdo inclui as tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que você usa. Para ter mais informações sobre a privacidade de dados, consulte as [Perguntas frequentes sobre privacidade de dados](#). Para ter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog [AWS Shared Responsibility Model and GDPR](#) no Blog de segurança da AWS.

Para fins de proteção de dados, recomendamos que você proteja as credenciais da Conta da AWS e configure as contas de usuário individuais com o AWS IAM Identity Center ou o AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos da AWS. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure o registro em log das atividades da API e do usuário com o AWS CloudTrail.
- Use as soluções de criptografia da AWS, juntamente com todos os controles de segurança padrão dos Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar a AWS por meio de uma interface de linha de comando ou uma API, use um endpoint do FIPS. Para ter mais informações sobre endpoints do FIPS, consulte [Federal Information Processing Standard \(FIPS\) 140-2](#).

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de email dos seus clientes, em marcações ou campos de formato livre, como um campo Name (Nome). Isso inclui quando você trabalha com o SDK for Java ou Serviços da AWS outro usando o console, a API AWS CLI AWS ou os SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendamos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Trabalhando com TLS no SDK for Java

O AWS SDK for Java usa os recursos de TLS de sua plataforma Java subjacente. [Neste tópico, mostramos exemplos usando a implementação do OpenJDK usada pelo Amazon Corretto 17.](#)

Para trabalhar com eleServiços da AWS, o JDK subjacente deve oferecer suporte a uma versão mínima do TLS 1.2, mas o TLS 1.3 é recomendado.

Os usuários devem consultar a documentação da plataforma Java que estão usando com o SDK para descobrir quais versões do TLS estão habilitadas por padrão e como habilitar e desabilitar versões específicas do TLS.

Como verificar as informações da versão do TLS

Usando o OpenJDK, o código a seguir mostra o uso do SSLContext para imprimir quais versões [de TLS/SSL](#) são suportadas.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocolNames()))
```

Por exemplo, o Amazon Corretto 17 (OpenJDK) produz a seguinte saída.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

Para ver o handshake SSL em ação, e qual versão do TLS é usada, você pode usar a propriedade do sistema javax.net.debug.

Por exemplo, execute aplicativos Java que usam TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

O aplicativo registra o handshake SSL semelhante ao seguinte.

```
...
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",
...

```

Aplique uma versão mínima do TLS

O SDK for Java sempre prefere a versão mais recente do TLS compatível com a plataforma e o serviço. Se você deseja aplicar uma versão mínima específica do TLS, consulte a documentação da sua plataforma Java.

Para JVMs baseadas em OpenJDK, você pode usar a propriedade do sistema.

`jdk.tls.client.protocols`

Por exemplo, se você quiser que os clientes do serviço SDK em seu aplicativo usem o TLS 1.2, mesmo que o TLS 1.3 esteja disponível, forneça a seguinte propriedade do sistema.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWSAtualização de endpoints de API para TLS 1.2

Consulte esta [postagem do blog](#) para obter informações sobre a migração dos endpoints de AWS API para o TLS 1.2 na versão mínima.

Gerenciamento de identidade e acesso

O AWS Identity and Access Management (IAM) é um serviço da AWS service (Serviço da AWS) que ajuda a controlar o acesso aos recursos da AWS de forma segura. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) a usar os recursos do AWS. O IAM é um AWS service (Serviço da AWS) que pode ser usado sem custo adicional.

Tópicos

- [Público](#)
- [Como autenticar com identidades](#)
- [Gerenciamento do acesso usando políticas](#)

- [Como Serviços da AWS funcionam com o IAM](#)
- [Solução de problemas de identidade e acesso do AWS](#)

Público

O uso do AWS Identity and Access Management (IAM) varia dependendo do trabalho que for realizado no AWS.

Usuário do serviço: se você usar Serviços da AWS para fazer seu trabalho, o administrador fornecerá as credenciais e as permissões necessárias. À medida que usar mais recursos do AWS para fazer seu trabalho, você poderá precisar de permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um atributo na AWS, consulte [Solução de problemas de identidade e acesso do AWS](#) ou o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Administrador do serviço – Se você for o responsável pelos recursos do AWS na empresa, provavelmente terá acesso total ao AWS. Cabe a você determinar quais funcionalidades e recursos do AWS os usuários do serviço devem acessar. Assim, é necessário enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os conceitos básicos do IAM. Para saber mais sobre como sua empresa pode usar o IAM com a AWS, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Administrador do IAM – Se você for um administrador do IAM, talvez queira saber detalhes sobre como pode gravar políticas para gerenciar o acesso ao AWS. Para visualizar exemplos de políticas baseadas em identidade da AWS que podem ser usadas no IAM, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Como autenticar com identidades

A autenticação é a forma como você faz login na AWS usando suas credenciais de identidade. É necessário ser autenticado (fazer login na AWS) como o usuário raiz da Usuário raiz da conta da AWS, como usuário do IAM ou assumindo um perfil do IAM.

Você pode fazer login na AWS como uma identidade federada usando credenciais fornecidas por uma fonte de identidades. AWS IAM Identity Center Os usuários do IAM Identity Center, a autenticação única da empresa e as suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como uma identidade federada, o administrador já

configurou anteriormente a federação de identidades usando perfis do IAM. Quando você acessa a AWS usando a federação, está indiretamente assumindo um perfil.

É possível fazer login no AWS Management Console ou no portal de acesso da AWS dependendo do tipo de usuário que você é. Para obter mais informações sobre como fazer login na AWS, consulte [How to sign in to your Conta da AWS](#) (Como fazer login na conta da) no Início de Sessão da AWS User Guide (Guia do usuário do).

Se você acessar a AWS programaticamente, a AWS fornecerá um kit de desenvolvimento de software (SDK) e uma interface da linha de comando (CLI) para você assinar criptograficamente as solicitações usando as suas credenciais. Se você não utilizar as ferramentas da AWS, deverá assinar as solicitações por conta própria. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte [Assinar solicitações de API da AWS](#) no Guia do usuário do IAM.

Independentemente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, a AWS recomenda o uso da autenticação multifator (MFA) para aumentar a segurança de sua conta. Para saber mais, consulte [Autenticação multifator](#) no GuiaAWS IAM Identity Center do usuário. [Usar a autenticação multifator \(MFA\) naAWS](#) no Guia do usuário do IAM.

Usuário root da Conta da AWS

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos os recursos e Serviços da AWS na conta. Essa identidade, denominada usuário raiz da Conta da AWS, é acessada por login com o endereço de email e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele pode executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte [Tarefas que exigem credenciais de usuário raiz](#) no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários, inclusive os que precisam de acesso de administrador, usem a federação com um provedor de identidades para acessar Serviços da AWS usando credenciais temporárias.

Identidade federada é um usuário de seu diretório de usuários corporativos, um provedor de identidades da web, o AWS Directory Service, o diretório do Centro de Identidade ou qualquer

usuário que acesse os Serviços da AWS usando credenciais fornecidas por meio de uma fonte de identidade. Quando as identidades federadas acessam Contas da AWS, elas assumem funções que fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no Centro de Identidade do IAM ou se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todas as suas Contas da AWS e aplicações. Para obter mais informações sobre o Centro de Identidade do IAM, consulte “[What is IAM Identity Center?](#)” (O que é o Centro de Identidade do IAM?) no Guia do usuário do AWS IAM Identity Center.

Grupos e usuários do IAM

Um [usuário do IAM](#) é uma identidade dentro da Conta da AWS que tem permissões específicas para uma única pessoa ou aplicação. Sempre que possível, recomendamos depender de credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte [Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo](#) no Guia do usuário do IAM.

Um [grupo do IAM](#) é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e atribuir a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de funções. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas uma função pode ser assumida por qualquer pessoa que precisar dela. Os usuários têm credenciais permanentes de longo prazo, mas as funções fornecem credenciais temporárias. Para saber mais, consulte [Quando criar um usuário do IAM \(em vez de uma função\)](#) no Guia do usuário do IAM.

Funções do IAM

Uma [função do IAM](#) é uma identidade dentro da Conta da AWS que tem permissões específicas. Ela é semelhante a um usuário do IAM, mas não está associada a uma pessoa específica. É possível assumir temporariamente um perfil do IAM no AWS Management Console [alternando perfis](#). É possível assumir um perfil chamando uma operação de API da AWS CLI ou da AWS, ou usando um

URL personalizado. Para obter mais informações sobre métodos para o uso de perfis, consulte [Usar perfis do IAM](#) no Guia do usuário do IAM.

Os perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, ela é associada ao perfil e recebe as permissões definidas pelo perfil. Para obter mais informações sobre perfis para federação, consulte [Criar um perfil para um provedor de identidades de terceiros](#) no Guia do usuário do IAM. Se você usar o IAM Identity Center, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte [Permission sets](#) (Conjuntos de permissões) no Guia do usuário do AWS IAM Identity Center.
- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode assumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (um principal confiável) em outra conta acesse recursos em sua conta. As funções são a principal forma de conceder acesso entre contas. No entanto, alguns Serviços da AWS permitem que você anexe uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recurso para acesso entre contas, consulte [Como as funções do IAM diferem das políticas baseadas em recurso](#) no Guia do usuário do IAM.
- Acesso entre serviços: alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando uma função de serviço ou uma função vinculada ao serviço.
 - Permissões de principal: ao usar um usuário ou uma função do IAM para executar ações na AWS, você é considerado um principal. As políticas concedem permissões a uma entidade principal. Quando você usa alguns serviços, pode executar uma ação que, em seguida, aciona outra ação em outro serviço. Nesse caso, você deve ter permissões para executar ambas as ações. Para ver se uma ação exige ações dependentes adicionais em uma política, consulte [Ações, recursos e chaves de condição para Serviços da AWS](#) na Referência de autorização do serviço.
 - Função de serviço: uma função de serviço é uma [função do IAM](#) que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um

perfil de serviço do IAM. Para obter mais informações, consulte [Criar um perfil para delegar permissões a um AWS service \(Serviço da AWS\)](#) no Guia do usuário do IAM.

- Função vinculada a serviço: uma função vinculada a serviço é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode assumir a função de executar uma ação em seu nome. Os perfis vinculados ao serviço aparecem em sua Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para funções vinculadas ao serviço.
- Aplicações em execução no Amazon EC2: é possível usar uma função do IAM para gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer solicitações da AWS CLI ou da AWS API. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma função da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém a função e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte [Usar uma função do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2](#) no Guia do usuário do IAM.

Para saber se deseja usar as funções do IAM, consulte [Quando criar uma função do IAM \(em vez de um usuário\)](#) no Guia do usuário do IAM.

Gerenciamento do acesso usando políticas

Você controla o acesso na AWS criando políticas e anexando-as a identidades ou recursos da AWS. Uma política é um objeto na AWS que, quando associado a uma identidade ou recurso, define suas permissões. A AWS avalia essas políticas quando uma entidade principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas são armazenadas na AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte [Visão geral das políticas JSON](#) no Guia do usuário do IAM.

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis, e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação, independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de funções do AWS Management Console, da AWS CLI ou da API da AWS.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou função do IAM. Essas políticas controlam quais ações os usuários e funções podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte [Criar políticas do IAM](#) no Guia do usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda mais como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou função. As políticas gerenciadas são políticas independentes que podem ser anexadas a vários usuários, grupos e funções na Conta da AWS. As políticas gerenciadas incluem políticas gerenciadas pela AWS e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recurso são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de função do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve [especificar um principal](#) em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, funções, usuários federados ou Serviços da AWS.

Políticas baseadas em recursos são políticas em linha que estão localizadas nesse serviço. Não é possível usar as políticas gerenciadas da AWS do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais principais (membros, usuários ou funções da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Amazon S3, AWS WAF e Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. Para saber mais sobre ACLs, consulte [Visão geral da lista de controle de acesso \(ACL\)](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

A AWS oferece suporte a tipos de política menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade e dos seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou a função no campo Principal não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte [Limites de permissões para identidades do IAM](#) no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs): SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (UO) no AWS Organizations. O AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS pertencentes à sua empresa. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades em contas-membro, incluindo cada Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizations e SCPs, consulte [Como os SCPs funcionam](#) no Guia do usuário do AWS Organizations.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para uma função ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou da função e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recurso. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina se deve permitir uma solicitação quando há vários tipos de política envolvidos, consulte [Lógica da avaliação de políticas](#) no Guia do usuário do IAM.

Como Serviços da AWS funcionam com o IAM

Para obter uma visão geral de como Serviços da AWS funcionam com a maioria dos atributos do IAM, consulte [Serviços da AWS compatíveis com o IAM](#) no Guia do usuário do IAM.

Para saber como usar um AWS service (Serviço da AWS) específico com o IAM, consulte a seção de segurança do Guia do usuário do serviço relevante.

Solução de problemas de identidade e acesso do AWS

Use as seguintes informações para ajudar a diagnosticar e corrigir problemas comuns que podem ser encontrados ao trabalhar com o AWS e o IAM.

Tópicos

- [Não tenho autorização para executar uma ação no AWS](#)
- [Não estou autorizado a realizar iam:PassRole](#)
- [Quero permitir que as pessoas fora da minha Conta da AWS acessem meus recursos do AWS](#)

Não tenho autorização para executar uma ação no AWS

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso `my-example-widget` fictício, mas não tem as permissões awes:`GetWidget` fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
awes:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso `my-example-widget` usando a ação `awes:GetWidget`.

Se você precisar de ajuda, entre em contato com seu administrador da AWS. Seu administrador é a pessoa que forneceu a você suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não tem autorização para executar a ação `iam:PassRole`, as suas políticas deverão ser atualizadas para permitir que você passe um perfil para o AWS.

Alguns Serviços da AWS permitem que você transmita um perfil existente para o serviço, em vez de criar um perfil de serviço ou um perfil vinculado ao serviço. Para fazer isso, um usuário deve ter permissões para passar o perfil para o serviço.

O erro de exemplo a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no AWS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar a função para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:  
iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador da AWS. Seu administrador é a pessoa que forneceu a você suas credenciais de login.

Quero permitir que as pessoas fora da minha Conta da AWS acessem meus recursos do AWS

Você pode criar uma função que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir a função. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte o seguinte:

- Para saber se o AWS oferece suporte a esses recursos, consulte [Como Serviços da AWS funcionam com o IAM](#).

- Para saber como conceder acesso a seus recursos em todas as Contas da AWS pertencentes a você, consulte [Fornecimento de acesso a um usuário do IAM em outra Conta da AWS pertencente a você](#) no Guia de usuário do IAM.
- Para saber como conceder acesso a seus recursos para terceiros Contas da AWS, consulte [Fornecimento de acesso a Contas da AWS pertencentes a terceiros](#) no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte [Conceder acesso a usuários autenticados externamente \(federação de identidades\)](#) no Guia do usuário do IAM.
- Para saber a diferença entre usar perfis e políticas baseadas em recursos para acesso entre contas, consulte [Como os perfis do IAM diferem de políticas baseadas em recursos](#) no Guia do usuário do IAM.

Validação de conformidade para este produto ou serviço da AWS

Para saber se um AWS service (Serviço da AWS) está no escopo de programas de conformidade específicos, consulte [Serviços da AWS no escopo por programa de conformidade](#) e selecione o programa de conformidade em que você está interessado. Para obter informações gerais, consulte [Programas de conformidade da AWS](#).

É possível fazer download de relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte [Downloading Reports in AWS Artifact](#).

Sua responsabilidade de conformidade ao usar o Serviços da AWS é determinada pela confidencialidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelos regulamentos e leis aplicáveis. A AWS fornece os seguintes recursos para ajudar com a conformidade:

- [Guias de início rápido de segurança e conformidade](#): estes guias de implantação discutem considerações sobre arquitetura e fornecem as etapas para a implantação de ambientes de linha de base focados em segurança e conformidade na AWS.
- [Architecting for HIPAA Security and Compliance on Amazon Web Services](#) (Arquitetura para segurança e conformidade com HIPAA no Amazon Web Services): esse whitepaper descreve como as empresas podem usar a AWS para criar aplicações adequadas aos padrões HIPAA.

Note

Nem todos os Serviços da AWS estão qualificados pela HIPAA. Para mais informações, consulte a [Referência dos serviços qualificados pela HIPAA](#).

- [Recursos de conformidade da AWS](#): essa coleção de manuais e guias pode ser aplicada a seu setor e local.
- [Avaliar recursos com regras](#) no AWS Config Developer Guide (Guia do desenvolvedor do CCI): o serviço AWS Config avalia como as configurações de recursos estão em conformidade com práticas internas, diretrizes do setor e regulamentos.
- [AWS Security Hub](#): este AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança na AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a [Referência de controles do Security Hub](#).
- [AWS Audit Manager](#): esse AWS service (Serviço da AWS) ajuda a auditar continuamente seu uso da AWS para simplificar a forma como você gerencia os riscos e a conformidade com regulamentos e padrões do setor.

Esse produto ou serviço da AWS segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) compatíveis. Para obter informações de segurança sobre o serviço da AWS, consulte a [página de documentação de segurança do serviço da AWS](#) e [Serviços da AWS que estão no escopo dos esforços de conformidade da AWS por programa de conformidade](#).

Resiliência para este produto ou serviço da AWS

A infraestrutura global da AWS é criada com base em Regiões da AWS e zonas de disponibilidade.

As Regiões da AWS fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, que são conectadas com baixa latência, throughputs elevados e redes altamente redundantes.

Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre regiões e zonas de disponibilidade da AWS, consulte [Infraestrutura global da AWS](#).

Esse produto ou serviço da AWS segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) compatíveis. Para obter informações de segurança sobre o serviço da AWS, consulte a [página de documentação de segurança do serviço da AWS](#) e [Serviços da AWS que estão no escopo dos esforços de conformidade da AWS por programa de conformidade](#).

Segurança da infraestrutura para esse produto ou serviço da AWS

Esse produto ou serviço da AWS usa serviços gerenciados e, portanto, é protegido pela segurança de rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte [Segurança na Nuvem AWS](#). Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança de infraestrutura, consulte [Proteção de infraestrutura](#) em Pilar segurança: AWS Well-Architected Framework.

Você usa chamadas de API publicadas pela AWS para acessar esse produto ou serviço da AWS pela rede. Os clientes devem oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o [AWS Security Token Service](#) (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Esse produto ou serviço da AWS segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) compatíveis. Para obter informações de segurança sobre o serviço da AWS, consulte a [página de documentação de segurança do serviço da AWS](#) e [Serviços da AWS que estão no escopo dos esforços de conformidade da AWS por programa de conformidade](#).

Migre da versão 1.x para a 2.x do AWS SDK for Java

O AWS SDK for Java 2.x é uma grande reescrita da base de código 1.x construída sobre o Java 8+. Ele inclui muitas atualizações, como melhor consistência, facilidade de uso e imutabilidade fortemente reforçada. Esta seção descreve os principais recursos que são novos na versão 2.x e fornece orientação sobre como migrar seu código da 1.x para a versão 2.x.

Tópicos

- [O que há de novo](#)
- [O que há de diferente entre AWS SDK for Java 1.x e 2.x](#)
- [Use o SDK for Java 1.x side-by-side](#)

O que há de novo

- Você também pode configurar seus próprios clientes HTTP. Consulte a [configuração do transporte HTTP](#).
- Os clientes assíncronos agora estão realmente sem bloqueio e retornam objetos `CompletableFuture`. Consulte Programação [assíncrona](#).
- As operações que retornam várias páginas possuem respostas autopaginadas. Dessa forma, você pode focar seu código no que fazer com a resposta, sem a necessidade de verificar e acessar as páginas subsequentes. Consulte [Paginação](#).
- O desempenho do horário de início do SDK para as funções do AWS Lambda foi aprimorado. Veja as [melhorias no desempenho do horário de início do SDK](#).
- A versão 2.x ofereça suporte a um novo método resumido para criar solicitações.

Example

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

Para obter mais detalhes sobre os novos recursos e ver exemplos de código específicos, consulte as outras seções deste guia.

- [Início rápido](#)
- [Configurando](#)

- [Exemplos de código para o AWS SDK for Java 2.x](#)
- [Use o SDK](#)
- [Segurança para o AWS SDK for Java](#)

O que há de diferente entre AWS SDK for Java 1.x e 2.x

Esta seção descreve as principais mudanças a serem observadas ao converter um aplicativo do uso da AWS SDK for Java versão 1.x para a versão 2.x.

Alteração do nome do pacote

Uma mudança notável do SDK for Java 1.x para o SDK for Java 2.x é a alteração do nome do pacote. Os nomes dos pacotes começam com `software.amazon.awssdk` no SDK 2.x, enquanto o SDK 1.x usa `com.amazonaws`.

Esses mesmos nomes diferenciam os artefatos do Maven do SDK 1.x ao SDK 2.x. Os artefatos do Maven para o SDK 2.x usam o GroupID, enquanto o SDK 1.x usa o `software.amazon.awssdk` GroupID. `com.amazonaws`

Algumas vezes, seu código exige uma `com.amazonaws` dependência para um projeto que, de outra forma, usa somente artefatos do SDK 2.x. Um exemplo disso é quando você trabalha com o lado do servidor AWS Lambda. Isso foi mostrado na seção [Configurar um projeto Apache Maven](#), anteriormente neste guia.

Note

Vários nomes de pacotes no SDK 1.x contêm `v2`. O uso de `v2` nesse caso, geralmente significa que o código no pacote é direcionado para funcionar com a versão 2 do serviço. Como o nome completo do pacote começa com `com.amazonaws`, esses são componentes do SDK 1.x. Exemplos desses nomes de pacotes no SDK 1.x são:

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

Adicionando a versão 2.x ao seu projeto

O Maven é a forma recomendada de gerenciar dependências ao usar o AWS SDK for Java 2.x. Para adicionar componentes da versão 2 ao seu projeto, atualize seu pom.xml arquivo com uma dependência do 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>
```

Criadores de clientes

Você deve criar todos os clientes usando o método do compilador de clientes. Construtores não estão mais disponíveis.

Example de criar um cliente na versão 1.x

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

Example de criar um cliente na versão 2.x

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

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

Configuração do cliente

Na versão 1.x, a configuração do cliente SDK foi modificada com a definição de uma `ClientConfiguration` instância no cliente ou no construtor do cliente. Na versão 2.x, a configuração do cliente é dividida em classes de configuração separadas. Com as classes de configuração separadas, você pode configurar clientes HTTP diferentes para clientes assíncronos versus clientes síncronos, mas ainda usar a mesma classe. `ClientOverrideConfiguration`

Example da configuração do cliente na versão 1.x

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

Example da configuração do cliente síncrono na versão 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 da configuração do cliente assíncrono na versão 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();
```

[Para um mapeamento completo dos métodos de configuração do cliente entre 1.x e 2.x, consulte o changelog AWS SDK for Java 2.x.](#)

Métodos setter

No AWS SDK for Java 2.x, os nomes dos métodos setter não incluem o prefixo `set` or `with`. Por exemplo, `*.withEndpoint()` é agora `*.endpoint()`.

Example de usar métodos de configuração na 1.x

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

Example de usar métodos de configuração na 2.x

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

Nomes de classes

Todos os nomes de classes de clientes agora estão totalmente em letras maiúsculas e não são mais prefixados por `Amazon`. Estas alterações estão alinhadas com os nomes usados no AWS CLI. Para obter uma lista completa das mudanças no nome do cliente, consulte o [AWS SDK for Java changelog 2.x](#).

Example dos nomes de classe na 1.x

```
AmazonDynamoDB
```

AWSACMPCAAAsyncClient

Example dos nomes de classe na 2.x

```
DynamoDbClient  
AcmAsyncClient
```

Classe de região

A versão 1.x do AWS SDK for Java tinha várias classes Region e Regions, tanto no pacote principal como em muitos dos pacotes de serviço. As classes Region e Regions na versão 2.x estão agora recolhidas em uma classe principal, Region.

Example Classes Region e Regions na 1.x

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

Example Classe de região na 2.x

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

Para obter mais detalhes sobre as mudanças relacionadas ao uso da Region classe, consulte [Alterações no nome da classe de região](#).

POJOs imutáveis

Os clientes e solicitação de operação e objetos de resposta agora são imutáveis e não podem ser alterados após a criação. Para reutilizar uma variável de solicitação ou resposta, você deve criar um novo objeto para atribuir a ela.

Example de atualizar um objeto de solicitação na 1.x

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

Example de atualizar um objeto de solicitação em 2.x

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

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

Operações de streaming

Operações de streaming, como os `putObject` métodos Amazon S3 `getObject` e, agora oferecem suporte a E/S sem bloqueio. Como resultado, os POJOs de solicitação e resposta não são mais usados `InputStream` como parâmetro. Em vez disso, o objeto de solicitação aceita `RequestBody`, que é um fluxo de bytes. O cliente assíncrono aceita `AsyncRequestBody`.

Example da `putObject` operação do Amazon S3 em 1.x

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

Example da `putObject` operação do Amazon S3 em 2.x

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

Em paralelo, o objeto de resposta aceita o `ResponseTransformer` para clientes síncronos e o `AsyncResponseTransformer` para clientes assíncronos.

Example da `getObject` operação do Amazon S3 em 1.x

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

Example da `getObject` operação do Amazon S3 em 2.x

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
```

```
ResponseTransformer.toFile(Paths.get("key")));
```

Alterações na exceção

Os nomes das classes de exceção, suas estruturas e seus relacionamentos também foram alterados. `software.amazon.awssdk.core.exception.SdkException` é a nova `Exception` classe base que todas as outras exceções estendem.

Para obter uma lista completa dos nomes das classes de exceção 2.x mapeados para as exceções 1.x, consulte [Alterações no nome da classe de exceção](#).

Mudanças específicas do serviço

Alterações no nome da operação do Amazon S3

Muitos dos nomes de operação do Amazon S3 cliente foram alterados na AWS SDK for Java versão 2.x. Na versão 1.x, o cliente do Amazon S3 não é gerado diretamente da API de serviço. Isso resulta em inconsistência entre as operações do SDK e API de serviço. Na versão 2.x, agora geramos o cliente do Amazon S3 para ser mais consistente com a API de serviço.

Example da operação do cliente do Amazon S3 na 1.x

```
changeObjectStorageClass
```

Example da operação do cliente do Amazon S3 na 2.x

```
copyObject
```

Example da operação do cliente do Amazon S3 na API de serviço do Amazon S3

```
CopyObject
```

[Para obter uma lista completa dos mapeamentos de nomes de operações, consulte o changelog AWS SDK for Java 2.x.](#)

Acesso entre regiões

Para as melhores práticas de segurança, o acesso entre regiões não é mais suportado para clientes individuais.

Na versão 1.x, serviços como Amazon S3, Amazon SNS, e Amazon SQS têm permissão para acessar recursos além dos limites da região. Isso não é mais permitido na versão 2.x usando-se o mesmo cliente. Se você precisar acessar um recurso em uma região diferente, deverá criar um cliente nessa região e recuperar o recurso usando o cliente apropriado.

Alterações adicionais do cliente

Este tópico descreve alterações adicionais no cliente padrão no AWS SDK for Java 2.x.

Alterações padrão do cliente

- A cadeia de fornecedores de credenciais padrão para o Amazon S3 não inclui mais credenciais anônimas. Você deve especificar o acesso anônimo ao Amazon S3 manualmente usando o `AnonymousCredentialsProvider`.
- As seguintes variáveis de ambiente relacionadas à criação do cliente padrão foram alteradas.

1.x	2.x
<code>AWS_CBOR_DISABLED</code>	<code>CBOR_ENABLED</code>
<code>AWS_ION_BINARY_DISABLE</code>	<code>BINARY_ION_ENABLED</code>

- As seguintes propriedades do sistema relacionadas à criação do cliente padrão foram alteradas.

1.x	2.x
<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>com.amazonaws.sdk.disableCbor</code>	<code>aws.cborEnabled</code>
<code>com.amazonaws.sdk.disableIonBinary</code>	<code>aws.binaryIonEnabled</code>

- As seguintes propriedades do sistema não são mais suportadas no 2.x.

1.x

com.amazonaws.sdk.disableCertChecking

com.amazonaws.sdk.enableDefaultMetrics

com.amazonaws.sdk.enableThrottledRetry

com.amazonaws.regions.RegionUtils.fileOverride

com.amazonaws.regions.RegionUtils.disableRemote

com.amazonaws.services.s3.disableImplicitGlobalClients

com.amazonaws.sdk.enableInRegionOptimizedMode

- A configuração de região de carregamento de um endpoints.json arquivo personalizado não é mais suportada.

Alterações no provedor de credenciais

Provedor de credenciais

Esta seção fornece um mapeamento das mudanças de nome das classes e métodos do provedor de credenciais entre as versões 1.x e 2.x do AWS SDK for Java. Ela também lista algumas das principais diferenças na forma como as credenciais são processadas pelo SDK na versão 2.x:

- O provedor de credenciais padrão carrega as propriedades do sistema antes das variáveis de ambiente na versão 2.x. Para obter mais informações, consulte [Usando credenciais](#).
- O método construtor é substituído pelos métodos `create` ou `builder`.

Example

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

- A atualização assíncrona não é mais definida por padrão. Você deve especificá-la com o builder do provedor de credenciais.

Example

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

- Você pode especificar um caminho para um arquivo de perfil personalizado usando o `ProfileCredentialsProvider.builder()`.

Example

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

- O formato do arquivo de perfil foi alterado para melhor corresponder à AWS CLI. Para obter detalhes, consulte [Configurando o AWS CLI](#) no Guia do AWS Command Line Interface Usuário.

Alterações do provedor de credenciais mapeadas entre as versões 1.x e 2.x

Alterações nos nomes dos métodos

1.x	2.x
<code>AWSCredentialsProvider.getCredentials</code>	<code>AwsCredentialsProvider.resolveCredentials</code>
<code>DefaultAWSCredentialsProviderChain.getInstance</code>	Sem suporte
<code>AWSCredentialsProvider.getInstance</code>	Não suportado
<code>AWSCredentialsProvider.refresh</code>	Não suportado

Alterações nos nomes das variáveis do ambiente

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

Alterações nos nomes de propriedades do sistema

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

Como renomear um nome de classe

Esta seção descreve as mudanças implementadas no AWS SDK for Java 2.x para usar as `Regions` classes `Region` e.

Configuração da região

- Alguns AWS serviços não têm endpoints específicos da região. Ao usar esses serviços, você deve definir a região como `Region.AWS_GLOBAL` ou `Region.AWS_CN_GLOBAL`.

Example

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

- As classes `com.amazonaws.regions.Regions` e `com.amazonaws.regions.Region` agora estão combinadas em uma classe `software.amazon.awssdk.regions.Region`.

Mapeamentos de nomes de métodos e classes

As tabelas a seguir mapeiam as classes relacionadas à região entre as versões 1.x e 2.x do AWS SDK for Java. Você pode criar uma instância dessas classes usando o método `of()`.

Example

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

Alterações nos métodos da classe de regiões

1.x	2.x
Regions.fromName	Region.of
Regions.getName	Region.id
Regions.getDescription	Não suportado
Regiões.getCurrentRegion	Não suportado
Regions.DEFAULT_REGION	Não suportado
Regions.name	Não suportado

Alterações nos métodos da classe de região

1.x	2.x
Region.getName	Region.id
Região.hasHttpsEndpoint	Não suportado
Região.hasHttpEndpoint	Não suportado
Região.getAvailableEndpoints	Não suportado
Region.createClient	Não suportado

RegionMetadata mudanças no método de classe

1.x	2.x
RegionMetadata.Obter getName	RegionMetadata.nome
RegionMetadata.obter domínio	RegionMetadata.domínio
RegionMetadata.Obter partição	RegionMetadata.partição

ServiceMetadata mudanças no método de classe

1.x	2.x
Região. getServiceEndpoint	ServiceMetadata.Endpoint para (região)
Região. isServiceSupported	ServiceMetadata.regions () .contains (Região)

Alterações nos nomes de classe de

Este tópico contém um mapeamento de alterações de nomes relacionados à classe de exceção entre as versões 1.x e 2.x.

Esta tabela mapeia as alterações no nome da classe de exceção.

1.x	2.x
com.amazonaws.SdkBaseException ion com.amazonaws.AmazonClientE xception	software.amazon.awssdk.core .exception.SdkException
com.amazonaws.SdkClientException	software.amazon.awssdk.core .exception.SdkClientException
com.amazonaws.AmazonService Exception	software.amazon.awssdk.awsc ore.exception AwsServiceExc ption

A tabela a seguir mapeia os métodos em classes de exceção entre as versões 1.x e 2.x.

1.x	2.x
AmazonServiceException.getRequestId	SdkServiceException.requestId
AmazonServiceException.getServiceName	AwsServiceException.awsErrorDetails().serviceName
AmazonServiceException.getErrorCode	AwsServiceException.awsErrorDetails().errorCode
AmazonServiceException.getErrorMessage	AwsServiceException.awsErrorDetails().errorMessage
AmazonServiceException.getStatusCode	AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode
AmazonServiceException.getHttpHeaders	AwsServiceException.awsErrorDetails().sdkHttpResponse().headers
AmazonServiceException.rawResponse	AwsServiceException.awsErrorDetails().rawResponse

Status de migração de bibliotecas e utilitários

SDK para bibliotecas e utilitários Java

A tabela a seguir lista o status da migração das bibliotecas e utilitários do SDK for Java.

Nome da versão 1.12.x	Nome da versão 2.x	Desde a versão em 2.x
DynamoDBMapper	DynamoDbEnhancedClient	2.12.0
Waiters	Waiters	2.15.0

Nome da versão 1.12.x	Nome da versão 2.x	Desde a versão em 2.x
CloudFrontUrlSigner, CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3 TransferManager	2.19.0
Cliente de metadados EC2	Cliente de metadados EC2	2.19.29
Analisador de URI S3	Analisador de URI S3	2.20.41
Criador de políticas do IAM	Criador de políticas do IAM	2.20.126
Buffer do lado do cliente do Amazon SQS	Envio automático de solicitações	ainda não lançado
Listeners de progresso	Listeners de progresso	ainda não lançado

Bibliotecas relacionadas

A tabela a seguir lista as bibliotecas lançadas separadamente, mas que funcionam com o SDK for Java 2.x.

Nome usado com a versão 2.x do SDK for Java	Desde a versão
Cliente de criptografia Amazon S3	3.0.0 1
AWSCliente de criptografia de banco de dados para DynamoDB	3.0.0 2

¹ O cliente de criptografia para o Amazon S3 está disponível usando a seguinte dependência do Maven.

```
<dependency>
    <groupId>software.amazon.encryption.s3</groupId>
    <artifactId>amazon-s3-encryption-client-java</artifactId>
    <version>3.x</version>
</dependency>
```

² O AWS Database Encryption Client para DynamoDB está disponível usando a seguinte dependência do Maven.

```
<dependency>
    <groupId>software.amazon.cryptography</groupId>
    <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
    <version>3.x</version>
</dependency>
```

Use o SDK for Java 1.x side-by-side

Você pode usar as duas versões do AWS SDK for Java em seus projetos.

A seguir, mostramos um exemplo dopom.xml arquivo de um projeto que usaAmazon S3 da versão 1.x eDynamoDB da versão 2.16.1.

Example Exemplo de POM

Este exemplo mostra uma entrada depom.xml arquivo para um projeto que usa as versões 1.x e 2.x do 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>
```

Histórico do documento

Este tópico descreve mudanças importantes no Guia do AWS SDK for Java Desenvolvedor ao longo de sua história.

Este guia foi publicado pela última vez em 17 de outubro de 2023.

Alteração	Descrição	Data
<u>???</u>	Adicione informações sobre o status de migração de bibliotecas e utilitários do SDK for Java v1.x para v2.x	17 de outubro de 2023
<u>???</u>	Atualizar o tópico de configuração do Gradle	17 de outubro de 2023
<u>the section called “Ignore atributos nulos de objetos aninhados”</u>	Adicione informações sobre a anotação do DynamoDB Enhanced Client. @DynamoDb IgnoreNulls	22 de setembro de 2023
<u>the section called “Acesso entre regiões”</u>	Adicione informações sobre o acesso entre regiões aos buckets do Amazon S3.	31 de agosto de 2023
<u>the section called “Preserve objetos vazios”</u>	Adicione uma seção que discuta a @DynamoDb PreserveEmptyObject anotação.	25 de agosto de 2023
<u>???</u>	Atualize a seção do cliente do serviço.	15 de agosto de 2023
<u>the section called “Recomendações do cliente”</u>	Desde a versão 0.23, o AWS CRT suporta sistemas operacionais baseados em musl, como o Alpine Linux.	11 de agosto de 2023

Alteração	Descrição	Data
	As recomendações do cliente HTTP agora refletem o suporte musl.	
<u>the section called “API IAM Policy Builder”</u>	Adicionar seção da API IAM Policy Builder	31 de julho de 2023
<u>the section called “Conceitos básicos”</u>	Corrija vários trechos na seção Get Started do tópico DynamoDB Enhanced Client.	24 de julho de 2023
<u>the section called “Suporte de proxy”</u>	Adicione informações e exemplos de suporte de proxy HTTP para cada cliente HTTP.	2 de junho de 2023
Reorganize o sumário	Promova a <u>Exemplos de código</u> seção e <u>Trabalhar com Serviços da AWS</u> as entradas de TOC de nível superior.	24 de maio de 2023
<u>the section called “Adicionar dependência de registro”</u>	Mostre as dependências do Gradle na seção de registro.	23 de maio de 2023
<u>the section called “Paginação”</u>	Atualize o tópico de paginação .	18 de maio de 2023
<u>the section called “Configurar um projeto Gradle”</u>	Atualize a configuração do projeto Gradle.	3 de maio de 2023
<u>API de cliente aprimorada do DynamoDB</u>	Lançado o tópico reescrito da DynamoDB Enhanced Client API.	28 de abril de 2023
<u>Atualize as instruções do tutorial de introdução</u>	Arquétipo do Maven modificado para incluir a opção CredentialsProvider; instruções modificadas de acordo.	11 de abril de 2023

Alteração	Descrição	Data
<u>the section called “Recomendações do cliente”</u>	Adicionar orientação de decisão do cliente HTTP	30 de março de 2023
Atualizações de práticas recomendadas do IAM	Guia atualizado para alinhamento com as práticas recomendadas do IAM. Para obter mais informações, consulte <u>Práticas recomendadas de segurança no IAM</u> .	14 de março de 2023
<u>the section called “Recarregar credenciais do perfil”</u>	Adicione uma seção sobre como recarregar as credenciais do perfil.	9 de fevereiro de 2023
<u>the section called “Configurar o cliente AWS HTTP baseado em CRT”</u>	Tópico de atualização para a versão GA.	8 de fevereiro de 2023
<u>the section called “Trabalhe com metadados da instância do Amazon EC2”</u>	Adicione um exemplo guiado do cliente Java SDK para o serviço de metadados da instância Amazon S3.	1 de fevereiro de 2023
<u>the section called “AWSClient S3 baseado em CRT”</u>	Adicione uma seção para o cliente AWS S3 baseado em CRT.	19 de dezembro de 2022
<u>the section called “Gerenciador de transferências S3”</u>	Atualize os exemplos do Amazon S3 Transfer Manager para a versão GA.	19 de dezembro de 2022
<u>the section called “Práticas recomendadas”</u>	Foi adicionada a seção de melhores práticas.	18 de novembro de 2022

Alteração	Descrição	Data
<u>the section called “Carregar credenciais temporárias de um processo externo”</u>	Foi adicionada uma seção sobre o carregamento de credenciais de um processo externo.	15 de novembro de 2022
<u>the section called “Métricas do cliente de serviço”</u>	Lista de métricas atualizada com o requisito de uso do cliente HTTP.	9 de novembro de 2022
<u>the section called “Gerenciador de transferências S3”</u>	Código de exemplo corrigido.	2 de novembro de 2022
<u>the section called “Reduza o tempo de inicialização do SDK para AWS Lambda”</u>	Seção atualizada com opções adicionais para reduzir o tempo de inicialização do Lambda.	1º de novembro de 2022
<u>the section called “Clientes HTTP”</u>	Foram adicionadas informações de configuração para cobrir todos os clientes HTTP no SDK.	26 de outubro de 2022
<u>the section called “Registro em log”</u>	Tópico de registro atualizado para incluir detalhes de registro de conexão para todos os clientes HTTP.	4 de outubro de 2022
<u>the section called “AWS serviços de banco de dados”</u>	Foi adicionada a seção de visão geral dos serviços de AWS banco de dados e do SDK for Java 2.x.	13 de setembro de 2022
<u>O EC2-Classic Networking está sendo descontinuado</u>	O EC2-Classic será descontinuado em 15 de agosto de 2022.	28 de julho de 2022

Alteração	Descrição	Data
<u>the section called “Opções adicionais de autenticação”</u>	A atualização da dependência é necessária para a autenticação de login único.	18 de julho de 2022
<u>the section called “Transport Layer Security (TLS)”</u>	Atualize as informações de segurança do TLS.	8 de abril de 2022
<u>the section called “Opções adicionais de autenticação”</u>	Foram adicionadas mais informações sobre como configurar e usar credenciais.	22 de fevereiro de 2021
<u>the section called “Configurar um projeto GraalVM Native Image”</u>	Novo tópico para configurar um projeto GraalVM Native Image.	18 de fevereiro de 2021
<u>the section called “Waiters”</u>	Garçons lançados; tópico adicionado para o novo recurso.	30 de setembro de 2020
<u>the section called “Métricas do SDK”</u>	Métricas lançadas; tópico adicionado para o novo recurso.	17 de agosto de 2020
<u>the section called “Amazon Pinpoint”, the section called “Amazon Cognito”, the section called “Amazon SNS”</u>	Foram adicionados exemplos de tópicos para Amazon PinpointAmazon Cognito, Amazon SNS e.	30 de maio de 2020
<u>the section called “Reduza o tempo de inicialização do SDK para AWS Lambda”</u>	Tópico de desempenho da AWS Lambda função adicionado.	29 de maio de 2020
<u>the section called “Definir o TTL da JVM para pesquisas de nomes DNS”</u>	Foi adicionado o tópico de cache de DNS do JVM TTL.	27 de abril de 2020

Alteração	Descrição	Data
<u>the section called “Configurar um projeto Apache Maven”, the section called “Configurar um projeto Gradle”</u>	Novos tópicos de configuração do Maven e do Gradle.	21 de abril de 2020
<u>the section called “Transport Layer Security (TLS)”</u>	O TLS 1.2 foi adicionado à seção de segurança.	19 de março de 2020
<u>the section called “Inscrever-se no Amazon Kinesis Data Streams”</u>	Exemplos de Kinesis stream adicionados.	2 de agosto de 2018
<u>the section called “Paginação”</u>	Adição do tópico de paginação automática.	5 de abril de 2018
<u>???</u>	Foram adicionados exemplos de tópicos para IAMAmazon EC2, CloudWatch DynamoDB e.	29 de dezembro de 2017
<u>the section called “Amazon S3”</u>	Foi adicionado um exemplo de getobjects paraAmazon S3.	7 de agosto de 2017
<u>the section called “Programação assíncrona”</u>	Adição do tópico assíncrono.	4 de agosto de 2017
Versão GA do <u>AWS SDK for Java2.x</u>	AWS SDK for Java versão 2 (v2) lançada.	28 de junho de 2017

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.