

Guia do Desenvolvedor para a versão 2.x

AWS SDK for Java 2.x



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

Copyright © 2024 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 clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não são propriedade da Amazon pertencem aos respectivos proprietários, os quais podem ou não ser afiliados, estar conectados ou ser patrocinados pela Amazon.

Table of Contents

Guia do desenvolvedor - AWS SDK for Java 2.x	1
Começar a usar o SDK	1
Desenvolver aplicativos móveis	1
Manutenção e suporte para as versões principais do SDK	1
Recursos adicionais	2
Contribuição para o SDK	2
Tutorial de conceitos básicos	3
Etapa 1: configurar para este tutorial	3
Etapa 2: criar o projeto	3
Etapa 3: escrever o código	8
Etapa 4: compilar e executar o aplicativo	12
Bem-sucedida	13
Limpeza	13
Próximas etapas	13
Configuração	15
Visão geral da configuração	15
Capacidade de login no portal de acesso AWS	16
Configurar acesso de logon único para o SDK	16
Faça login usando o AWS CLI	17
Instale o Java e uma ferramenta de compilação	18
Opções de autenticação adicionais	18
Configurar um projeto do Apache Maven	18
Pré-requisitos	19
Criar um projeto Maven	19
Configurar o compilador Java para Maven	20
Declarar o SDK como dependência	21
Definir dependências para módulos do SDK	22
Compilar o projeto	24
Configurar um projeto do Gradle	25
Configurar um projeto do GraalVM Native Image	31
Pré-requisitos	31
Crie um projeto usando o arquétipo	32
Crie uma imagem nativa	33
Usar o SDK	34

Trabalhar com os clientes de serviço	34
Crie um cliente de serviço	34
Configuração padrão de cliente	35
Configurar a página de serviço	35
Fazer solicitações.	36
Manipulador de resposta	36
Feche o cliente de serviço	37
Tratamento de exceções	37
Como usar waiters	38
Clientes HTTP	39
Repetições	39
Tempos limite	39
Interceptores de execução	40
Mais informações	40
Fornecimento de credenciais temporárias ao SDK	40
Configurar o acesso às credenciais temporárias	40
Cadeia de fornecedores de credenciais padrão	42
Usar um provedor de credenciais ou uma cadeia de fornecedores específicos	44
Usar perfis	45
Carregar credenciais temporárias de um processo externo	48
Fornecer credenciais temporárias em código	51
Leia as credenciais da função do IAM em Amazon EC2	55
Use Regiões da AWS	56
Configurar explicitamente uma Região da AWS	57
Determinar a região a partir do ambiente	57
Conferir a disponibilidade de serviço	59
Escolher um endpoint específico	59
Reduza o tempo de inicialização do SDK para AWS Lambda	59
Usar o <code>URLConnectionHttpClient</code> do SDK	60
Usar o <code>AwsCrtAsyncHttpClient</code> do SDK	60
Remover dependências não utilizadas do cliente HTTP	61
Configurar clientes de serviço para reduzir as pesquisas	62
Inicializar o cliente SDK fora do manipulador da função do Lambda	63
Minimize a injeção de dependência	64
Use uma mira do Maven Archetype AWS Lambda	64
Lambda SnapStart	64

Alterações na versão 2.x que afetam o tempo de inicialização	64
Recursos adicionais do	64
Cientes HTTP	65
Clientes disponíveis	65
Recomendações do cliente	66
Padrões inteligentes	69
Suporte de proxy	71
Configurar o cliente HTTP baseado em Apache	73
Configurar o cliente HTTP baseado em URLConnection	78
Configurar o cliente HTTP baseado em Netty	84
Configurar os clientes HTTP baseados em AWS CRT	90
Tratamento de exceções	101
Por que exceções desmarcadas?	101
AwsServiceException (e subclasses)	101
SdkClientException	102
Exceções e comportamento de nova tentativa	102
Registro em log	103
Arquivo de configuração do Log4j 2	103
Adicionar dependência de registro	103
Erros e avisos específicos do SDK	104
Registro em log do resumo de requisição/resposta	105
Registro em log detalhado	106
Definir o JVM TTL para pesquisas de nome DNS	111
Como definir o TTL da JVM	112
Melhores práticas	112
Reutilizar um cliente do SDK	113
Fechar streams de entrada	113
Ajustar as configurações de HTTP	113
Usar o OpenSSL para o Netty	114
Configurar tempos limite da API	114
Usar métricas	115
Usar recursos do SDK	116
Recursos gerais	116
Recursos específicos do serviço	116
Trabalhar com resultados paginados	116
Paginação síncrona	117

Paginação assíncrona	120
Pesquisar estados de recursos	125
Pré-requisitos	125
Usar agentes de espera	126
Configurar waiters	127
Exemplos de código	128
Usar programação assíncrona	128
Operações sem streaming	128
Operações de streaming	131
Operações avançadas	135
Trabalhar com o HTTP/2	136
Usar métricas do SDK	137
Pré-requisitos	137
Como ativar a coleta de métricas	138
Quando as métricas estão disponíveis?	139
Quais informações são coletadas?	139
Como posso usar essas informações?	140
Métricas de cliente de serviço	140
Trabalhe com Serviços da AWS	145
CloudWatch	145
Obtenha métricas de CloudWatch	146
Publicar dados de métrica personalizada no CloudWatch	148
Trabalho com alarmes do CloudWatch	150
Use os CloudWatch eventos da Amazon	154
serviços de banco de dados AWS	158
Amazon DynamoDB	158
Amazon RDS	159
Amazon Redshift	159
Amazon Aurora Sem Servidor v1	159
Amazon DocumentDB	160
DynamoDB	160
Trabalhe com tabelas em DynamoDB	160
Trabalhar com itens no DynamoDB	170
Associar objetos a itens do DynamoDB	177
Amazon EC2	297
Gerenciar instâncias Amazon EC2	297

Zonas Regiões da AWS de uso e disponibilidade	304
Trabalhe com grupos de segurança em Amazon EC2	307
Trabalhar com metadados da instância do Amazon EC2	312
IAM	318
Gerenciar chaves de IAM acesso	319
Gerenciar usuários do IAM	325
Criar políticas do IAM	329
Trabalhar com políticas do IAM	337
Trabalhe com certificados de IAM servidor	343
Kinesis	348
Inscrever-se no Amazon Kinesis Data Streams	348
Lambda	359
Invocar uma função do Lambda	359
Listar funções do Lambda	360
Excluir uma função do Lambda	361
Amazon S3	362
Usar pontos de acesso ou pontos de acesso multirregionais	363
Operações de buckets	364
Operações com objetos	372
URLs pré-assinados	381
Acesso entre regiões	390
Somos de verificação	391
Usar um cliente do S3 de alta performance	393
Transferir arquivos e diretórios	396
Amazon SNS	404
Criar um tópico	404
Listar seus tópicos do Amazon SNS	405
Inscrever um endpoint em um tópico	406
Publicar uma mensagem em um tópico	407
Cancelar a inscrição de um endpoint de um tópico	408
Excluir um tópico	409
Amazon SQS	410
Operações de fila	410
Operações de mensagens	414
Amazon Transcribe	417
Configurar o microfone	417

Criar um publicador	418
Criar o cliente e iniciar o streaming	421
Mais informações	417
Exemplos de código	423
Ações e cenários	423
API Gateway	425
Application Auto Scaling	429
Application Recovery Controller	438
Aurora	441
Auto Scaling	476
Amazon Bedrock	539
Amazon Bedrock Runtime	544
CloudFront	570
CloudWatch	590
CloudWatch Eventos	641
CloudWatch Registros	647
Amazon Cognito Identity	658
Provedor de identidade do Amazon Cognito	666
Amazon Comprehend	693
DynamoDB	704
Amazon EC2	771
Amazon ECS	842
Elastic Load Balancing	856
MediaStore	901
OpenSearch Serviço	916
EventBridge	925
Previsão	957
AWS Glue	970
HealthImaging	994
IAM	1020
AWS IoT	1106
AWS IoT data	1133
Amazon Keyspaces	1136
Kinesis	1162
AWS KMS	1175
Lambda	1193

MediaConvert	1218
Migration Hub	1240
Amazon Personalize	1254
Eventos do Amazon Personalize	1284
Runtime do Amazon Personalize	1287
Amazon Pinpoint	1292
API de SMS e voz do Amazon Pinpoint	1337
Amazon Polly	1341
Amazon RDS	1347
Amazon Redshift	1387
Amazon Rekognition	1393
Registro de domínio do Route 53	1461
Amazon S3	1484
S3 Glacier	1591
SageMaker	1607
Secrets Manager	1636
Amazon SES	1649
API v2 do Amazon SES	1662
Amazon SNS	1666
Amazon SQS	1715
Step Functions	1736
AWS STS	1760
AWS Support	1763
Systems Manager	1786
Amazon Textract	1796
Amazon Transcribe	1806
Exemplos entre serviços	1823
Criar uma aplicação para enviar dados para uma tabela do DynamoDB	1823
Criar um chatbot Amazon Lex	1824
Criação de uma aplicação do Amazon SNS	1824
Criar um aplicativo de mensagem	1825
Criar uma aplicação com tecnologia sem servidor para gerenciar fotos	1825
Criar uma aplicação Web para monitorar dados do DynamoDB	1826
Criar uma aplicação Web para rastrear dados do Amazon Redshift	1826
Crie um rastreador de itens de trabalho do Aurora Sem Servidor	1826
Criar uma aplicação para analisar o feedback dos clientes	1827

Detectar EPI em imagens	1828
Detectar objetos em imagens	1828
Detectar pessoas e objetos em um vídeo	1829
Publicar mensagens em filas	1829
Usar o API Gateway para invocar uma função do Lambda	1830
Usar Step Functions para invocar funções do Lambda	1830
Usar eventos programados para invocar uma função do Lambda	1831
Segurança	1832
Proteção de dados	1832
Transport Layer Security (TLS)	1834
Verificar versões do TLS	1834
Aplicar versões do TLS	1835
Migrar para TLS 1.2	1835
Identity and Access Management	1835
Público	1836
Autenticando com identidades	1836
Gerenciamento do acesso usando políticas	1840
Como Serviços da AWS trabalhar com o IAM	1843
Solução de problemas AWS de identidade e acesso	1843
Compliance Validation	1845
Resiliência	1846
Infrastructure Security	1847
Migrar para a versão 2	1849
Novidades da versão 2	1849
tep-by-step Instruções S	1850
Visão geral das etapas	1850
Exemplo de migração	1851
O que é diferente entre o 1.x e o 2.x	1862
Alteração do nome do pacote	1863
Adicionar a versão 2.x ao seu projeto	1863
POJOs imutáveis	1864
Métodos setter e getter	1865
Nomes de classes de modelo	1865
Bibliotecas e utilitários	1866
Alterações de cliente	1867
Alterações no provedor de credenciais	1912

Mudanças na região	1920
Alterações nas operações, solicitações e respostas	1922
Alterações na exceção	1923
Alterações na serialização	1925
Alterações específicas do serviço	1926
Alterações no arquivo de perfil	1931
Configuração externa	1933
Waiters	1936
Gerenciador de transferências do S3	1940
Utilitário de metadados do EC2	1946
CloudFront pré-assinando	1954
Análise de URI do S3	1957
Use o SDK para Java 1.x e 2.x lado a lado	1960
Chave OpenPGP	1962
Chave atual	1962
Histórico do documento	1964
.....	mcm1xxi

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 compilar aplicativos Java que funcionem com Amazon S3, Amazon EC2, DynamoDB e muito mais.

O AWS SDK for Java 2.x é uma reescrita principal do código de base da versão 1.x. Ele foi criado com base no Java 8+ e adiciona vários recursos frequentemente solicitados. Entre eles, suporte para E/S sem bloqueio e 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](#).

Começar a usar o SDK

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

Consulte [Configuração](#) para configurar seu ambiente de desenvolvimento.

Se você estiver atualmente 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, DynamoDB, Amazon EC2 e outros Serviços da AWS, consulte [Usar o SDK for Java](#) e [trabalhar com Serviços da AWS](#).

Desenvolver aplicativos móveis

Se você é um desenvolvedor de aplicativos móveis, o Amazon Web Services oferece a estrutura [AWS Amplify](#).

Manutenção e suporte para as versões principais do SDK

Para obter informações sobre manutenção e suporte para versões principais do SDK e suas dependências subjacentes, consulte os seguintes tópicos em [Guia de referência de SDKs e ferramentas da AWS](#):

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

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

Recursos adicionais

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

- [Referência da API do AWS SDK for Java 2.x](#)
- [Blog de desenvolvedor Java](#)
- [Tópico de desenvolvimento Java no AWS re:Post](#)
- [Origem do SDK](#) no GitHub
- [Biblioteca de exemplos de códigos do SDK da AWS](#)
- [@awsforjava](#) (Twitter)

Contribuição para o SDK

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

- Enviar problemas no GitHub:
 - [Enviar problemas com a documentação do guia do desenvolvedor](#)
 - [Enviar problemas do SDK](#)
- Participar de um bate-papo informal sobre o SDK no [canal gitter](#) do AWS SDK for Java 2.x

Conceitos básicos do AWS SDK for Java 2.x

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

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

Siga estas etapas para concluir este tutorial:

- [Etapa 1: configurar para este tutorial](#)
- [Etapa 2: criar o projeto](#)
- [Etapa 3: escrever o código](#)
- [Etapa 4: compilar e executar o aplicativo](#)

Etapa 1: configurar para este tutorial

Antes de iniciar este tutorial, é necessário instalar o seguinte:

- Permissão para acessar o Amazon S3
- Um ambiente de desenvolvimento Java configurado para acessar Serviços da AWS usando logon único ao AWS IAM Identity Center

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

Etapa 2: criar o projeto

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

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

2. Insira 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 informar
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 informando Y ou reinsira valores informando N.

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

O Maven cria a árvore de diretórios a seguir.

```
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 do arquivo de projeto `pom.xml`.

pom.xml

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

```
<?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>ssoidc</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 classe `App` criada pelo Maven. O método `main` é o ponto de entrada no aplicativo, que cria uma instância da classe `Handler` e, em seguida, chama seu método `sendRequest`.

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 classe `DependencyFactory` criada pelo Maven contém o método de fábrica `s3Client`, que cria e retorna uma instância do [S3Client](#). A instância do `S3Client` usa uma instância do cliente HTTP baseado em Apache. Isso ocorre porque você especificou `apache-client` quando o Maven solicitou o cliente HTTP a ser usado.

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

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

Classe **Handler**, 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 classe `Handler` pelo código a seguir. O método `sendRequest` é preenchido e as importações necessárias são adicionadas.

Classe **Handler**, implementada

Primeiro, o código cria um novo bucket do S3 com a última parte do nome gerada usando `System.currentTimeMillis()` para tornar o nome do bucket exclusivo.

Depois de criar o bucket no método `createBucket()`, o programa carrega um objeto usando o método [putObject](#) de `S3Client`. O conteúdo do objeto é uma string simples criada com o método `RequestBody.fromString`.

Por fim, o programa exclui o objeto seguido pelo bucket no método `cleanUp`.

```
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 classe `Handler` completa, crie e execute o aplicativo.

1. É essencial ter uma sessão ativa do IAM Identity Center. Para fazer isso, execute o comando `aws sts get-caller-identity` da AWS Command Line Interface e verifique a resposta. Se você ainda não tiver uma sessão ativa, consulte [esta seção](#) para obter as instruções.
2. Abra uma janela de terminal ou prompt de comando e navegue até o diretório `getstarted` do seu 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 bucket e objeto que o programa cria, execute as etapas a seguir.

1. Em `Handler.java`, comente a linha `cleanup(s3Client, bucket, key)` no método `sendRequest` e salve o arquivo.
2. Reconstrua o projeto executando `mvn clean package`.
3. Execute `mvn exec:java -Dexec.mainClass="org.example.App"` novamente 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 construiu com sucesso o primeiro aplicativo Java usando o SDK for Java 2.x.

Limpeza

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

- Se você ainda não tiver feito isso, no [console do S3](#), exclua todos os objetos e buckets criados quando você executou o aplicativo.
- Exclua a pasta do projeto (`getstarted`).

Próximas etapas

Agora que você entendeu o básico, poderá descobrir mais sobre o seguinte:

- [Trabalhar com Amazon S3](#)

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

Configure o AWS SDK for Java 2.x

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

Visão geral da configuração

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

- Você deve conseguir [entrar no portal de acesso da AWS](#) disponível em 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 PowerUserAccess AWS gerenciada 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 arquivo `credentials` contém credenciais temporárias.
 - Uma [instalação do Java 8](#) ou posterior.
 - Uma [ferramenta de automação de compilação](#), 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. O [AWS Toolkit para IntelliJ](#) e o [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 identidades externo que funciona independentemente do IAM Identity Center, descubra como você pode obter credenciais temporárias para o SDK para Java usar. Siga [estas instruções](#) para adicionar credenciais temporárias ao arquivo `~/.aws/credentials`.

Se seu provedor de identidade adicionar credenciais temporárias automaticamente ao arquivo `~/.aws/credentials`, certifique-se de que o nome do perfil seja `[default]` para que você não precise fornecer um nome de perfil ao SDK ou à 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 da URL é `d-xxxxxxxxx.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 acesso de logon único para o SDK

Depois de concluir a etapa 2 na [seção de acesso programático](#) para que o SDK use a autenticação do Centro de Identidade do IAM, o sistema deve conter os elementos a seguir.

- O AWS CLI, que você usa para iniciar uma [sessão do portal de AWS acesso](#) antes de executar seu aplicativo.
- Um arquivo `~/.aws/config` que contém um [perfil padrão](#). O SDK para Java usa as configurações do provedor de token de SSO do perfil para adquirir as credenciais antes de enviar solicitações à AWS. O `sso_role_name` valor, que é uma função do IAM conectada a um conjunto de permissões do IAM Identity Center, deve permitir o acesso aos Serviços da AWS usado em seu aplicativo.

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

```
[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 de SSO, consulte [Configuração do provedor de token de SSO](#) no Guia de Referência de SDKs e Ferramentas da AWS .

Se seu ambiente de desenvolvimento não estiver configurado para acesso programático conforme mostrado anteriormente, siga a [Etapa 2 no Guia de Referência de 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 AWS CLI comando 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 AWS acesso e executá-laaws sso login, não será necessário fornecer credenciais.

No entanto, você verá uma caixa de diálogo solicitando permissão para que o botocore acesse suas informações. O botocore é a base para a AWS CLI .

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

Instale o Java e uma ferramenta de compilação

Seu ambiente de desenvolvimento necessita 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 Adoptium.](#)
- 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 <https://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/>.

Opções de autenticação adicionais

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.

Configurar um projeto do Apache Maven

O [Apache Maven](#) pode ser usado para configurar e criar projetos do AWS SDK for Java ou [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 do Amazon Corretto 8 ou Amazon Corretto 11 mais recente da [página Corretto](#).
- 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 com a linha de comando, execute o seguinte comando em um terminal ou janela de 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 do Maven usando o kit de ferramentas de modelo do arquétipo. O arquétipo gera a estrutura básica para um projeto manipulador de funções do AWS Lambda. Esse arquétipo de projeto é pré-configurado para compilar com Java SE 8 e inclui uma dependência para a versão do SDK para Java 2.x especificada por `-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 de projeto do AWS Lambda conforme descrito anteriormente, a configuração do compilador de Java já está feita.

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 de projeto conforme descrito anteriormente, a versão mais recente do SDK já está configurada como uma dependência no projeto.

O arquétipo gera uma dependência de artefato da BOM (lista de materiais) para o ID do grupo `software.amazon.awssdk`. Com uma 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 arquivo `pom.xml` 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.

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

Se você criou o projeto usando o arquétipo de projeto conforme descrito anteriormente, ele já está configurado com várias dependências. Isso inclui dependências para manipuladores de funções do AWS Lambda 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 exemplo pom.xml acima, as dependências são de groupIds diferentes. A dependência s3 é de software.amazon.awssdk, enquanto a dependência aws-lambda-java-core é de com.amazonaws. A configuração do gerenciamento de dependências da BOM afeta os artefatos para software.amazon.awssdk, então é necessária uma versão para o artefato aws-lambda-java-core.

Para o desenvolvimento de manipuladores de função do Lambda usando o SDK para Java 2.x, a dependência correta é aws-lambda-java-core. No entanto, se seu aplicativo precisar gerenciar recursos do Lambda usando operações como listFunctions, deleteFunction, invokeFunction e createFunction, seu aplicativo exigirá a dependência a seguir.

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

Note

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

Adicione os módulos ao projeto do AWS service (Serviço da AWS) e recursos necessários para ele. Os módulos (dependências) que são gerenciados pela BOM do AWS SDK for Java 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 tiver interesse nas dependências do serviço do DynamoDB, consulte [este exemplo](#) no [Repositório de exemplos de código da AWS](#) no GitHub. (Procure o arquivo `pom.xml` 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 do Gradle

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

As etapas iniciais no exemplo a seguir vêm do [Guia de conceitos básicos do Gradle](#) para a versão 8.4. O uso de uma versão diferente pode gerar pequenas variações nos resultados.

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

1. Crie um diretório para armazenar seu projeto. Neste exemplo, o nome do diretório é demo.
2. Dentro do diretório demo, execute o comando `gradle init` e forneça os valores destacados em vermelho, conforme mostrado na saída da linha de comando a seguir. Para a demonstração, foi escolhido o Kotlin como a linguagem DSL do script de construção, mas um exemplo completo do Groovy também é mostrado no fim 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. Após a conclusão da tarefa `init`, o diretório `demo` 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 arquivo `build.gradle.kts` contém o seguinte conteúdo gerado automaticamente.

```
/*
 * 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 gerado automaticamente como base para seu projeto da AWS.
 - a. Para gerenciar as dependências do SDK para o projeto do Gradle, adicione a lista de materiais (BOM) do Maven do AWS SDK for Java 2.x para a seção `dependencies` 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

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

- b. Especifique os módulos do SDK que seu aplicativo precisa na seção `dependencies`. O exemplo a seguir 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 corrige automaticamente a versão correta das dependências declaradas usando as informações da 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 as próximas etapas, consulte o Guia de Conceitos Básicos no site do Gradle para obter instruções sobre como [criar e executar um aplicativo do Gradle](#).

Configurar um projeto do GraalVM Native Image para o AWS SDK for Java

Com as versões 2.16.1 e posteriores, o AWS SDK for Java fornece suporte imediato para as aplicações do GraalVM Native Image. Use o arquétipo `archetype-app-quickstart` do Maven para configurar um projeto com suporte de imagem nativa integrado.

Pré-requisitos

- Conclua as etapas em [Configuração do AWS SDK for Java 2.x](#).

- Instale o [GraalVM Native Image](#).

Crie um projeto usando o arquétipo

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

Note

Substitua `com.example.mynativeimageapp` pelo namespace do pacote completo do seu aplicativo. Substitua `mynativeimageapp` pelo nome do 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 do Maven configurado com dependências para o AWS SDK for Java, o Amazon S3 e o cliente HTTP ApacheHttpClient. Ele também inclui uma dependência para o [plug-in GraalVM Native Image do Maven](#), para que você possa criar imagens nativas usando o Maven.

Para incluir dependências para uma Amazon Web Services diferente, defina o valor do parâmetro `-Dservice` 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 parâmetro `-DhttpClient` como `netty-nio-client`. Para usar `URLConnectionHttpClient` como cliente HTTP síncrono em vez de `apache-client`, defina o parâmetro `-DhttpClient` como `url-connection-client`.

Crie uma imagem nativa

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

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

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

Use o AWS SDK for Java 2.x

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

Trabalhar com os clientes de serviço

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 método `builder()` retorna um objeto `builder` que permite personalizar o cliente de 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. Após configurar as propriedades desejadas, você poderá chamar o método `build()` para criar o cliente.

Como exemplo, o trecho de código a seguir instancia um objeto `Ec2Client` como um 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.

Não é necessário especificar o `Region` no Service Client Builder 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. Consulte a [seleção da região da AWS](#) para obter mais informações.

Configuração padrão de 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 a Região da AWS. Se as credenciais ou a região não puderem ser determinadas pelo ambiente no qual o aplicativo estiver em execução, a chamada a `create` falhará. Consulte [Como usar credenciais](#) e [Seleção de região](#) para obter mais informações sobre como o SDK determina as credenciais e a região a ser usada.

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

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

Configurar a página de serviço

Para personalizar as configurações de um cliente de serviço, use os métodos de configuração no método de fábrica `builder()`. Por conveniência e para criar um código mais legível, encadeie os métodos para definir múltiplas opções de configuração.

O exemplo a seguir mostra um `S3Client` que está 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();
```

Fazer solicitações.

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

Por exemplo, este trecho de código mostra como criar um objeto `RunInstancesRequest` 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.
RunInstancesResponse response = ec2Client.runInstances(runInstancesRequest);
```

Em vez de criar uma solicitação e transmitir a instância, o SDK fornece criadores que você pode usar para criar uma solicitação. Com um criador, é possível usar expressões do lambda em Java para criar a solicitação “em linha”.

O exemplo a seguir reescreve o exemplo anterior usando a versão de `runInstances` [método que usa um criador](#) para criar a solicitação.

```
// Create the request by using a lambda expression.
RunInstancesResponse response = ec2.runInstances(r -> r
    .imageId(amiId)
    .instanceType(InstanceType.T1_MICRO)
    .maxCount(1)
    .minCount(1));
```

Manipulador de resposta

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

Por exemplo, este trecho de código mostra como criar um objeto `RunInstancesResponse` para lidar com a resposta do Amazon EC2 imprimindo a `instanceId` para a nova instância a partir 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 múltiplas chamadas de serviço de API durante a vida 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.

Chame o método `close()` quando o cliente de serviço não for mais necessário para liberar recursos.

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

Se precisar de um cliente de serviço para uso único, você pode instanciar o cliente de serviço como um recurso em uma declaração `try`-com recursos. Os clientes de serviço implementam a interface [Autoclosable](#), então o JDK chama automaticamente o método `close()` no fim da instrução.

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();
    }
}
```

Tratamento de exceções

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

Uma [SdkServiceException](#), ou uma de suas subclasses, é a forma mais comum de exceção que o SDK lançará. Essas exceções representam respostas do serviço da AWS. Você também pode

lidar com uma [SdkClientException](#), que ocorre quando há um problema no lado do cliente (ou seja, em seu ambiente de desenvolvimento ou do 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 o Amazon S3. O código de exemplo envolve as exceções de cliente e de servidor, registra os detalhes e sai do 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);
}
```

Para obter mais informações, consulte [Tratamento de exceções](#).

Como usar waiters

Algumas solicitações demoram para serem processadas, como criar uma nova tabela DynamoDB ou criar um novo Amazon S3 bucket. Para garantir que o recurso esteja pronto antes que seu código continue sendo executado, use um Waiter.

Por exemplo, esse trecho de código cria uma nova tabela (“MyTable”) em DynamoDB, espera que a tabela 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);
```

Para obter mais informações, consulte [Como usar waiters](#).

Clientes HTTP

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

Repetições

Você pode alterar as configurações padrão para novas tentativas em seus clientes de serviço, incluindo o modo de nova tentativa e a estratégia de recuo. Para obter mais informações, consulte [a RetryPolicy classe](#) na Referência da AWS SDK for Java API.

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

Tempos limite

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

Para obter mais informações, consulte [apiCallTimeout](#) e [apiCallAttemptTimeout](#) na Referência AWS SDK for Java da API.

Interceptores de execução

Você pode escrever um código que intercepte a execução das solicitações e respostas de API em diferentes partes do ciclo de vida delas. 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 [a `ExecutionInterceptor` interface](#) na Referência AWS SDK for Java da API.

Mais informações

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

Fornecimento de credenciais temporárias ao SDK

Antes de fazer uma solicitação para o 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 aborda várias maneiras de habilitar o SDK para acessar credenciais temporárias.

Tópicos

- [Configurar o acesso às credenciais temporárias](#)
- [Cadeia de fornecedores de credenciais padrão](#)
- [Usar um provedor de credenciais ou uma cadeia de fornecedores específicos](#)
- [Usar perfis](#)
- [Carregar credenciais temporárias de um processo externo](#)
- [Fornecer credenciais temporárias em código](#)
- [Leia as credenciais da função do IAM em Amazon EC2](#)

Configurar o acesso às credenciais temporárias

Para aumentar a segurança, a AWS recomenda que você configure o SDK para Java para [usar credenciais temporárias](#) em vez de credenciais de longa duração. As credenciais temporárias consistem em uma chave de acesso (ID de chave de acesso e chave de acesso secreta) e um token

de sessão. Recomendamos que você [configure o SDK](#) para obter automaticamente credenciais temporárias, pois o processo de atualização do token é automático. No entanto, você pode [fornecer credenciais temporárias diretamente ao SDK](#).

Configuração do IAM Identity Center

Quando você configura o SDK para usar o acesso de login único do IAM Identity Center, conforme descrito em [???](#) neste guia, o SDK usa automaticamente credenciais temporárias.

O SDK usa o token de acesso do IAM Identity Center para obter acesso ao perfil do IAM que está configurado com a configuração `sso_role_name` em seu arquivo `config`. O SDK assume esse perfil do IAM e recupera credenciais temporárias para usar nas solicitações AWS service (Serviço da AWS).

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

Recuperar do portal de acesso AWS

Como alternativa à configuração de autenticação única do IAM Identity Center, você pode copiar e usar credenciais temporárias disponíveis no portal de acesso AWS. 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. [Criar um arquivo de credenciais compartilhadas](#)
2. No arquivo de credenciais, cole o texto do espaço reservado a seguir 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. Agora, o arquivo `~/.aws/credentials` 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. [Fazer login no portal de acesso da AWS](#)


```
.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 [SystemPropertyCredentialsProvider](#) classe para carregar credenciais temporárias das propriedades `aws.accessKeyId`, `aws.secretAccessKey`, e do sistema `aws.sessionToken` Java.

Note

Para mais informações sobre como definir as propriedades de sistema do Java, consulte o tutorial [Propriedades do sistema](#) no site Tutoriais do Java oficial.

2. Variáveis de ambiente

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

3. Token de identidade da web do AWS Security Token Service

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

4. Os arquivos `credentials` e `config` compartilhados

- O SDK usa o [ProfileCredentialsProvider](#) para carregar as configurações de login único do IAM Identity Center ou credenciais temporárias do [default] perfil nos arquivos compartilhados `credentials` e `config`.

O Guia de referência de AWS 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

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

5. Amazon ECS credenciais de contêiner

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

6. Amazon EC2 credenciais fornecidas pelo perfil do IAM da instância

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

Usar um provedor de credenciais ou uma cadeia de provedores específicos

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 para o `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 provedores, consulte Todas as classes de implementação conhecidas em [AwsCredentialsProvider](#).

Note

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

Usar perfis

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

O trecho de código a seguir demonstra como criar um cliente de serviço que usa as configurações definidas como parte do perfil nomeado `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 perfil `[default]` como padrão para seu aplicativo, defina a variável de ambiente `AWS_PROFILE` com 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 a propriedade do sistema `aws.profile` Java com o nome do perfil.

Recarregar credenciais de perfil

Você pode configurar qualquer provedor de credenciais que tenha um método `profileFile()` 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` e `aws_session_token`. Configurações como `region`, `sso_session`, `sso_account_id` e `source_profile` são ignoradas.

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

```
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 uma das

[várias implementações de conveniência](#) do `ProfileFileSupplier` fornecidas pelo SDK. Se seu caso de uso exigir, você pode fornecer sua própria implementação.

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

As configurações serão recarregadas quando `resolveCredentials()` for 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 método `ProfileFileSupplier.aggregate()` mescla o conteúdo de vários arquivos de configuração. Você decide se um arquivo é recarregado por chamada para o `resolveCredentials()` ou se as configurações de um arquivo são fixadas 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 pela variável `credentialsFilePath` sempre que `resolveCredentials()` é chamado e as configurações são alteradas. As configurações do objeto `profileFile` permanecem as mesmas.

```
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. Isto pode ser potencialmente perigoso, portanto, prossiga com cuidado. Outros provedores de credenciais devem ser preferidos, se possível. Ao usar esta opção, certifique-se de que o `config` arquivo esteja o mais bloqueado possível usando as melhores práticas de segurança para seu sistema operacional.

Certifique-se de que sua ferramenta de credenciais personalizadas não grave nenhuma informação secreta em `StdErr`. Os SDKs e AWS CLI podem capturar e registrar essas informações, possivelmente 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.

Usar a configuração **credential_process**

Se você tiver um método que forneça credenciais temporárias, poderá integrá-lo adicionando a configuração `credential_process` como parte de uma definição de perfil no arquivo `config`. 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 determinado e, em seguida, lê 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 nomeado `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 a [seção de credenciais do processo](#) no Guia de referência de AWS SDKs e ferramentas.

Usar a **ProcessCredentialsProvider**

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

Os exemplos a seguir mostram várias versões de como especificar um processo externo usando o `ProcessCredentialsProvider` e 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();
```

Fornecer credenciais temporárias em código

Se a cadeia de credenciais padrão, um provedor personalizado ou específico ou a cadeia de fornecedores não funcionar para seu aplicativo, você poderá fornecer as credenciais temporárias diretamente no código. Elas podem ser [credenciais de perfil do IAM](#), conforme [descrito acima](#), ou credenciais temporárias recuperadas do AWS Security Token Service (AWS STS). Se você recuperou credenciais temporárias usando o AWS STS, forneça-as a um cliente AWS service (Serviço da AWS) conforme mostrado no exemplo de código a seguir.

1. Assuma um perfil chamando `StsClient.assumeRole()`.
2. Crie um [StaticCredentialsProvider](#) objeto e forneça o `AwsSessionCredentials` objeto a ele.

3. Configure o construtor de cliente de serviço com o `StaticCredentialsProvider` e construa o cliente.

O exemplo a seguir cria um cliente de serviço do Amazon S3 usando credenciais temporárias retornadas pelo AWS STS para uma função assumida do 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);
}
}
```

Conjuntos de permissões

O seguinte conjunto de permissões definido no AWS IAM Identity Center permite que a identidade (usuário) execute as duas operações a seguir

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

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

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


Função assumida

Política de permissões do perfil assumido

A política de permissões a seguir é anexada ao perfil assumido no exemplo anterior. Essa política de permissões permite listar todos os buckets na mesma conta do perfil.

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

Política de confiança do perfil assumido

A política de confiança a seguir é anexada ao perfil assumido no exemplo anterior. A política permite que o perfil seja assumido 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": {}
    }
  ]
}
```

```
}
```

Leia as credenciais da função do IAM em Amazon EC2

Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo solicitações AWS CLI de AWS API. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte [Usar 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 configurar seu aplicativo Java para ser executado em uma instância do EC2 e permitir que o SDK for Java IAM adquira credenciais de função.

Adquira credenciais de função do IAM do ambiente

Se seu aplicativo criar um cliente AWS de serviço usando o `create` método (ou `builder().build()` métodos), o SDK for Java usará a cadeia de fornecedores de credenciais padrão. A cadeia de fornecedores de credenciais padrão pesquisa o ambiente de execução em busca de elementos de configuração que o SDK possa trocar por credenciais temporárias. A seção [the section called “Cadeia de fornecedores de credenciais padrão”](#) descreve o processo completo de pesquisa.

A etapa final na cadeia de fornecedores padrão está disponível somente quando seu aplicativo é executado em uma Amazon EC2 instância. Nesta etapa, o SDK usa um `InstanceProfileCredentialsProvider` para ler o perfil do IAM definido no perfil da instância do EC2. Depois, o SDK adquire credenciais temporárias para esse perfil do IAM.

Embora essas credenciais sejam temporárias e acabem expirando, o `InstanceProfileCredentialsProvider` as atualiza periodicamente para você, de maneira que elas continuem permitindo o acesso à AWS.

Adquira credenciais de função do IAM de forma programática

Como alternativa à cadeia de provedores de credenciais padrão que eventualmente usa um `InstanceProfileCredentialsProvider` no EC2, você pode configurar um cliente de serviço

explicitamente com um `InstanceProfileCredentialsProvider`. Essa abordagem é mostrada no trecho a seguir.

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

Adquira com segurança as credenciais da função do IAM

Por padrão, as instâncias do EC2 executam o [IMDS](#) (Instance Metadata Service), que permite que os SDKs acessem informações como `InstanceProfileCredentialsProvider` a função do IAM que foi configurada. As instâncias do EC2 executam duas versões do IMDS por padrão:

- Serviço de metadados da instância versão 1 (IMDSv1) – um método de solicitação/resposta
- Serviço de metadados da instância versão 2 (IMDSv2): um método orientado a sessões

[O IMDSv2 é uma abordagem mais segura](#) do que o IMDSv1.

Por padrão, o Java SDK primeiro tenta o IMDSv2 para obter a função do IAM, mas se isso falhar, ele tenta o IMDSv1. No entanto, como o IMDSv1 é menos seguro, AWS recomenda usar somente o IMDSv2 e impedir que o SDK experimente o IMDSv1.

Para usar a abordagem mais segura, desabilite o SDK de usar o IMDSv1 fornecendo uma das configurações a seguir com um valor de `true`

- Variável de ambiente: `AWS_EC2_METADATA_V1_DISABLED`
- Propriedade do sistema JVM: `aws.disableEc2MetadataV1`
- Configuração do arquivo de configuração compartilhado: `ec2_metadata_v1_disabled`

Com uma dessas configurações definida como `true`, o SDK não carrega as credenciais da função IMDS usando o IMDSv1 se a chamada inicial do IMDSv2 falhar.

Use Regiões da AWS

Regiões da AWS permitir que os clientes do serviço acessem o Serviços da AWS que reside fisicamente em uma área geográfica específica.

Configurar explicitamente uma Região da AWS

Para definir explicitamente uma região, recomendamos usar as constantes definidas na classe [Região](#). Esta é uma enumeração de todas as regiões disponíveis publicamente.

Para criar um cliente com uma região enumerada da classe, use o método `region` do criador do cliente.

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

Se a região que você deseja usar não for uma das enumerações na classe `Region`, será possível criar uma região usando o método `of` estático. Esse recurso permite acessar as novas regiões sem atualizar o SDK.

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

Note

Depois de criar um cliente com o construtor, ele é imutável e Região da AWS não pode ser alterado. Se precisar trabalhar com várias Regiões da AWS para o mesmo serviço, crie vários clientes, um por região.

Deixar o SDK determinar automaticamente a região pelo ambiente

Quando seu código é executado em Amazon EC2 ou AWS Lambda, talvez você queira configurar os clientes para usarem o mesmo em Região da AWS que seu código está sendo executado. Isso separa seu código do ambiente em que ele está sendo executado e facilita a implantação de seu aplicativo em várias Regiões da AWS para reduzir a latência ou a redundância.

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

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

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

Entender a cadeia de provedores de região padrão

O SDK segue as seguintes etapas para procurar uma Região da AWS :

1. Qualquer região explícita definida usando-se `region` no criador 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

O Lambda contêiner define essa variável de ambiente.

3. O SDK verifica o arquivo de configuração AWS compartilhado e o arquivo de credenciais compartilhados (geralmente localizados em `~/.aws/config` e `~/.aws/credentials`). Se a `region` propriedade estiver presente, o SDK a usará.
 - Se o SDK encontrar a `region` propriedade nos dois arquivos do mesmo perfil (incluindo o `default` perfil), o SDK usará o valor no arquivo de credenciais compartilhado.
 - A variável de ambiente `AWS_CONFIG_FILE` pode ser usada para personalizar o local do arquivo de configuração compartilhado.
 - A variável de ambiente `AWS_PROFILE` ou a propriedade do sistema `aws.profile` podem ser usadas para especificar o perfil carregado pelo SDK.
4. O SDK tenta usar o serviço de metadados da Amazon EC2 instância (IMDS) para determinar a região da instância em execução no momento. Amazon EC2
 - Para maior segurança, você deve impedir que o SDK tente usar a versão 1 do IMDS. Você usa a mesma configuração para desativar a versão 1 descrita na [the section called “Com segurança”](#) seção.
5. Se, a essa altura, o SDK ainda não tiver encontrado uma região, a criação do cliente falhará com uma exceção.

Ao desenvolver AWS aplicativos, uma abordagem comum é usar o arquivo de configuração compartilhado (descrito em [Ordem de recuperação de credenciais](#)) para definir a região para o desenvolvimento local e confiar na cadeia de fornecedores da região padrão para determinar a

região quando o aplicativo é executado na AWS infraestrutura. Isso simplifica muito a criação do cliente e mantém a portabilidade do aplicativo.

Conferir a disponibilidade do serviço em uma região

Para ver se um determinado AWS service (Serviço da AWS) está disponível em uma região, use o `region` método `serviceMetadata` and no cliente de serviço.

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

Consulte a documentação [da classe Region](#) para saber o Regiões da AWS que você pode especificar e usar o prefixo do endpoint do serviço para consultar.

Escolher um endpoint específico

Em determinadas situações, como testar os recursos de pré-visualização de um serviço antes que os recursos se tornem disponíveis ao público em geral, talvez seja necessário especificar um endpoint em uma região. Nessas situações, os clientes de serviço podem ser configurados chamando o método `endpointOverride`.

Por exemplo, para configurar um Amazon EC2 cliente para usar a região da Europa (Irlanda) com um endpoint específico, 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](#) para ver a lista atual de regiões e seus endpoints correspondentes para todos os AWS serviços.

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 AWS Lambda das 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.

Usar o `URLConnectionHttpClient` do SDK

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

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

Note

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

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

- [Várias `Update*` operações](#) na AWS Security Hub API e também a [BatchUpdateFindings](#) operação
- Todas as [operações `Update*`](#) da 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 para o AWS service (Serviço da AWS) que você está usando. Verifique se as operações necessárias usam a operação PATCH.

Usar o `AwsCrtAsyncHttpClient` do SDK

Esse [AwsCrtAsyncHttpClient](#) é a contrapartida assíncrona para reduzir o tempo de inicialização do Lambda no SDK.

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

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

Remover dependências não utilizadas do cliente HTTP

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

```
<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 do `url-connection-client` por uma dependência do `aws-crt-client`.

Note

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

Configurar clientes de serviço para reduzir as pesquisas

Especificar uma região

Ao criar um cliente de serviço, chame o método `region` no builder do cliente de serviço. Isso reduz o [processo de pesquisa de região](#) padrão do SDK, que verifica as Região da AWS informações em vários locais.

Para manter o código Lambda independente da região, use o código a seguir dentro do método `region`. Esse código acessa a variável de ambiente `AWS_REGION` 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](#) quando você cria 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 em outros sistemas.

O trecho de código a seguir mostra um cliente de serviço do 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();
```

Inicializar o cliente SDK fora do manipulador da função do Lambda

Recomendamos inicializar um cliente SDK fora do método manipulador do 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, a instância `S3Client` é inicializada no construtor usando um método estático de fábrica. Se o contêiner gerenciado pelo ambiente Lambda for reutilizado, a instância `S3Client` inicializada será 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. Elas 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 o [Dagger](#).

Use uma mira do Maven Archetype AWS Lambda

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

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

Considere o Lambda SnapStart para Java

Se seus requisitos de tempo de execução forem compatíveis, AWS oferece o [Lambda SnapStart para Java](#). O Lambda SnapStart é uma solução baseada em infraestrutura que melhora o desempenho de inicialização das funções Java. Quando você publica uma nova versão de uma função, o Lambda a SnapStart inicializa e tira um instantâneo imutável e criptografado da memória e do estado do disco. SnapStart em 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 das bibliotecas [java.time](#) para objetos de data e hora, que faz parte do JDK
- Uso de [Slf4j](#) para uma fachada de registro em log

Recursos adicionais do

O Guia do AWS Lambda desenvolvedor contém uma [seção sobre as melhores práticas](#) para o desenvolvimento de funções Lambda que não são específicas de Java.

Para ver um exemplo de criação de um aplicativo nativo da nuvem em Java que usa AWS Lambda, consulte o conteúdo deste [workshop](#). O workshop discute a otimização do desempenho e outras práticas recomendadas.

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 for Java 2.x e o Maven para [criar uma imagem nativa do GraalVM](#).

Cientes HTTP

É possível alterar o cliente HTTP a ser usado para o cliente de serviço, bem como alterar a configuração padrão para clientes HTTP com o AWS SDK for Java 2.x. Esta seção discute os clientes HTTP e as configurações do SDK.

Cientes HTTP disponíveis no SDK para Java

Cientes síncronos

Cientes HTTP síncronos no SDK para Java implementam a interface [SdkHttpClient](#). Um cliente de serviço síncrono, como o `S3Client` ou o `DynamoDbClient`, requer o uso de um cliente HTTP síncrono. O AWS SDK for Java oferece três clientes HTTP síncronos.

ApacheHttpClient (padrão)

[ApacheHttpClient](#) é o cliente HTTP padrão para clientes de serviços síncronos. Para obter mais informações sobre a configuração do `ApacheHttpClient`, consulte [Configurar o cliente HTTP baseado em Apache](#).

AwsCrtHttpClient

O [AwsCrtHttpClient](#) fornece alto throughput e E/S sem bloqueio. Ele é desenvolvido com base no cliente HTTP AWS Common Runtime (CRT). Para receber informações sobre como configurar o `AwsCrtHttpClient` e usá-lo com clientes de serviço, consulte [the section called “Configurar os clientes HTTP baseados em AWS CRT”](#).

URLConnectionHttpClient

Para minimizar o número de jars e bibliotecas de terceiros utilizados pela aplicação, é possível usar o [URLConnectionHttpClient](#). Para obter mais informações sobre a configuração do `URLConnectionHttpClient`, consulte [Configurar o cliente HTTP baseado em URLConnection](#).

Clientes assíncronos

Clientes HTTP assíncronos no SDK para Java implementam a interface [SdkAsyncHttpClient](#). 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 síncronos.

NettyNioAsyncHttpClient (padrão)

[NettyNioAsyncHttpClient](#) é o cliente HTTP padrão usado por clientes assíncronos. Para obter mais informações sobre a configuração do `NettyNioAsyncHttpClient`, consulte [the section called “Configurar o cliente HTTP baseado em Netty”](#).

AwsCrtAsyncHttpClient

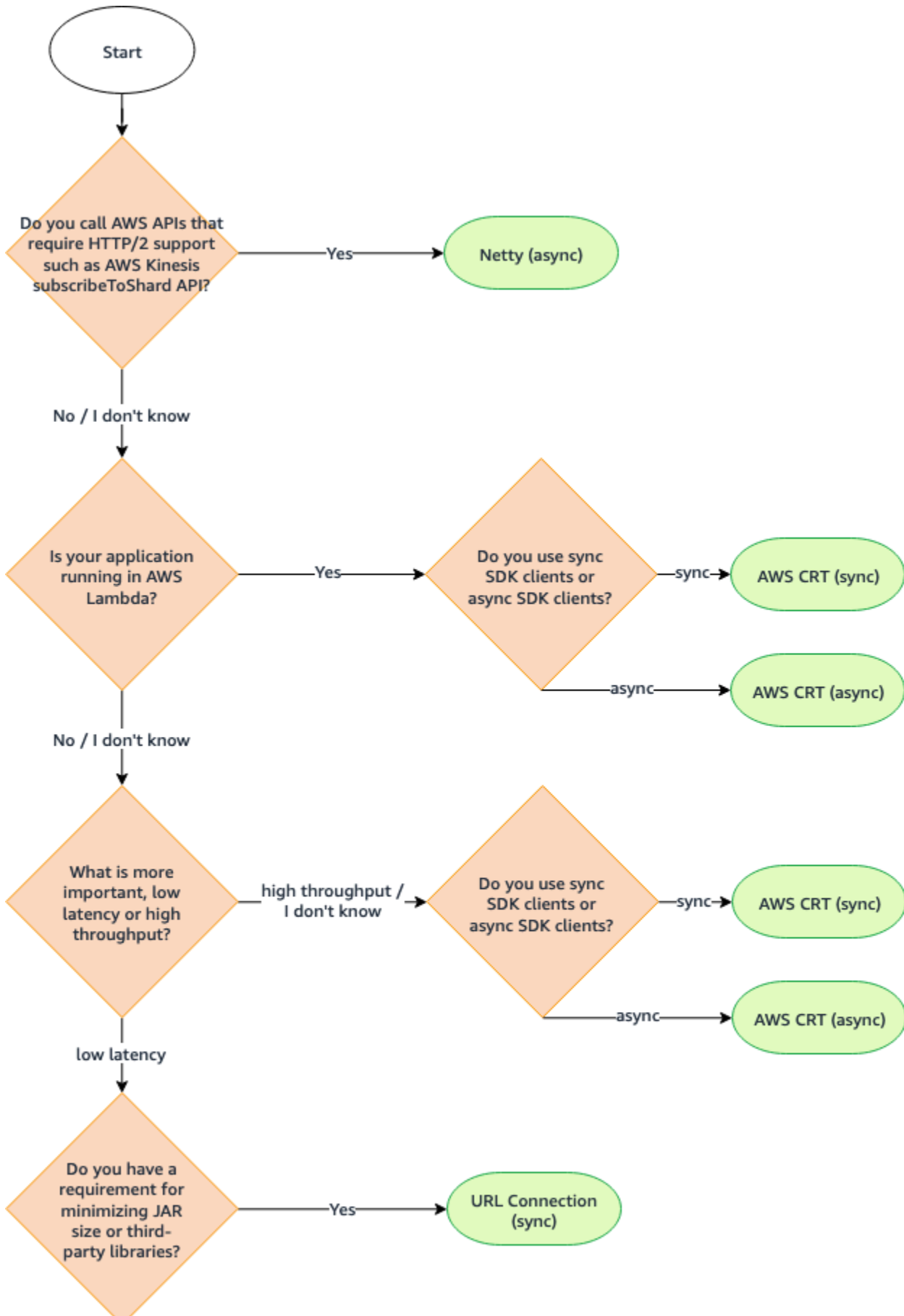
O [AwsCrtAsyncHttpClient](#) é baseado no cliente HTTP do AWS Common Runtime (CRT). Para obter mais informações sobre a configuração do `AwsCrtAsyncHttpClient`, consulte [the section called “Configurar os clientes HTTP baseados em AWS CRT”](#).

Recomendações do cliente HTTP

Diversos fatores entram em cena 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 contém informações detalhadas para cada cliente HTTP.

Cliente HTTP	Síncrono ou assíncrono	Quando usar	Limitação/desvantagem
Cliente HTTP baseado em Apache (cliente HTTP síncrono padrão)	Síncrono	Use-o se você preferir baixa latência em vez de alta taxa de transferência (throughput)	Tempo de inicialização mais lento em comparação com outros clientes HTTP
Cliente HTTP baseado em URLConnection	Síncrono	Use-o se você tiver um requisito rígido de limitar dependências de terceiros	Não é compatível com o método HTTP PATCH, exigido por algumas APIs, como as operações Update do Amazon APIGateway
Cliente HTTP síncrono baseado em AWS CRT ¹	Síncrono	<ul style="list-style-type: none"> • Use-o se seu aplicativo estiver sendo executado no AWS Lambda • Use-o se você preferir alta taxa de transferência (throughput) em vez de baixa latência • Use-o se preferir clientes do SDK síncronos. 	N/D
Cliente HTTP baseado em Netty	Assíncrono	• Use-o se seu aplicativo invocar APIs que exijam suporte a HTTP/2, como a API SubscribeToShard da Kinesis	Tempo de inicialização mais lento em comparação com

Cliente HTTP	Síncrono ou assíncrono	Quando usar	Limitação/desvantagem
(cliente HTTP síncrono padrão)			outros clientes HTTP
Cliente HTTP assíncrono baseado em AWS CRT ¹	Assíncrono	<ul style="list-style-type: none"> • Use-o se seu aplicativo estiver sendo executado no AWS Lambda • Use-o se você preferir alta taxa de transferência (throughput) em vez de baixa latência • Use-o se preferir clientes do SDK assíncronos. 	<ul style="list-style-type: none"> • Não oferece suporte a clientes de serviços que exigem suporte HTTP/2, como KinesisAsyncClient e TranscribeStreamingAsyncClient

¹Devido aos benefícios adicionais, recomendamos que você use os clientes HTTP baseados em AWS CRT, se possível.

Padrões de configuração inteligentes

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

Os padrões de configuração inteligentes definem valores razoáveis para as propriedades `connectTimeoutInMillis` e `tlsNegotiationTimeoutInMillis` com base em um valor de modo padrão fornecido por você. Você escolhe o valor de modo padrão com base nas características do seu aplicativo.

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

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

Service client

Use o builder 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 a propriedade `aws.defaultsMode` do sistema para especificar o modo padrão. Se você definir a propriedade do sistema em Java, precisará defini-la 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 definir o modo padrão como auto usando uma opção `-D` do comando `java`.

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

Environment variable

Defina um valor para a variável de ambiente `AWS_DEFAULTS_MODE` para selecionar o modo padrão para o 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 uma propriedade de configuração `defaults_mode` ao arquivo compartilhado `config` da AWS, conforme mostrado no exemplo a seguir.

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

Se você definir o modo padrão globalmente com a propriedade do sistema, a variável de ambiente ou o arquivo `config` da AWS, poderá substituir as configurações ao criar um cliente HTTP.

Quando você cria um cliente HTTP com o método `httpClientBuilder()`, 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` e `tlsNegotiationTimeoutInMillis`.

Suporte de proxy

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

Você configura proxies no código com um builder `ProxyConfiguration` específico do cliente ao criar o cliente de serviço. A seção de cada cliente HTTP neste tópico mostra um exemplo de configuração de proxy. Este [exemplo é para o cliente HTTP do Apache](#).

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

Propriedades do sistema	Descrição	Suporte do cliente HTTP
<code>http.proxyHost</code>	Nome do host do servidor proxy HTTP	Todos
<code>http.proxyPort</code>	Número da porta do servidor proxy HTTP	Todos
<code>http.proxyUser</code>	Nome de usuário para autenticação de proxy HTTP	Todos

Propriedades do sistema	Descrição	Suporte do cliente HTTP
http.proxyPassword	Senha para autenticação de proxy HTTP	Todos
http.nonProxyHosts	Lista de hosts que devem ser acessados diretamente, contornando o proxy. Também é válido quando o HTTPS é usado.	Todos
https.proxyHost	Nome do host do servidor proxy HTTPS	Netty, CRT
https.proxyPort	Número da porta do servidor proxy HTTPS	Netty, CRT
https.proxyUser	Nome de usuário para autenticação de proxy HTTPS	Netty, CRT
https.proxyPassword	Senha para autenticação de proxy HTTPS	Netty, CRT

Os termos usados nas tabelas significam:

- Tudo: todos os clientes HTTP oferecidos pelo SDK–`URLConnectionHttpClient`, `ApacheHttpClient`, `NettyNioAsyncHttpClient`, `AwsCrtAsyncHttpClient`
- Netty: o cliente HTTP baseado em Netty (`NettyNioAsyncHttpClient`)
- CRT: os clientes HTTP baseados em AWS CRT, (`AwsCrtHttpClient` e `AwsCrtAsyncHttpClient`)

Você pode usar uma combinação de configuração do cliente HTTP e propriedades do sistema. O builder `ProxyConfiguration` de cada cliente HTTP fornece um valor de `useSystemPropertyValues`. Por padrão, esse valor é `true`. Quando o valor é `true`, o SDK usa automaticamente valores de propriedade do sistema para opções que não são fornecidas usando o builder `ProxyConfiguration`.

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 a propriedade `endpoint` no código, conforme mostrado no trecho de código anterior, você pode usar as propriedades do sistema a seguir.

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

Configurar o cliente HTTP baseado em Apache

Por padrão, os clientes de serviço síncrono no AWS SDK for Java 2.x usam um cliente HTTP baseado em Apache, o [ApacheHttpClient](#). O `ApacheHttpClient` do SDK é baseado no [HttpClient](#) do Apache.

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

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

Acesse o `ApacheHttpClient`

Na maioria das situações, você usa o `ApacheHttpClient` sem nenhuma configuração explícita. Você vai declarar os clientes de serviço e o SDK vai configurar o `ApacheHttpClient` com valores padrão para você.

Se você quiser configurar o `ApacheHttpClient` explicitamente ou usá-lo com vários clientes de serviço, precisará disponibilizá-lo para configuração.

Nenhuma configuração 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 classe `ApacheHttpClient` disponível para o código em runtime, mas não no momento da compilação. Se você não estiver configurando o cliente HTTP baseado em Apache, não precisará especificar uma dependência para ele.

No seguinte trecho XML de um arquivo `pom.xml` do Maven, a dependência declarada com `<artifactId>s3</artifactId>` traz 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>
```

```
</dependencies>
```

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

Configuração necessária

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

Consulte o exemplo a seguir de um arquivo `pom.xml` do 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>
```

Usar e configurar o `ApacheHttpClient`

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

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

Prática recomendada: dedicar uma instância do **ApacheHttpClient** a um cliente de serviço

Se você precisar configurar uma instância do `ApacheHttpClient`, recomendamos que você crie a instância `ApacheHttpClient` dedicada. Faça isso usando o método `httpClientBuilder` do builder do cliente do 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 instância do `ApacheHttpClient` não for fechada quando não for mais necessária.

O exemplo a seguir cria um `S3Client` e configura a instância embutida do `ApacheHttpClient` com os valores `maxConnections` e `connectionTimeout`. A instância HTTP é criada usando o método `httpClientBuilder` do `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 instância do **ApacheHttpClient**

Para ajudar a reduzir o uso de recursos e memória do 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 instância do `ApacheHttpClient` 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 instância `ApacheHttpClient` configurada é transmitida ao método `httpClient` de cada construtor. Quando os clientes do serviço e o cliente HTTP não são mais necessários, o código os fecha explicitamente. O código fecha o cliente HTTP 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.builder()
    .maxConnections(100).build();

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

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

// Perform work with the s3Client and dynamoDbClient.

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


Exemplo de configuração do proxy

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

```
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 trecho da linha de comando a seguir.

```
$ 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 do proxy HTTPS.

Configurar o cliente HTTP baseado em URLConnection

O AWS SDK for Java 2.x oferece um cliente HTTP [URLConnectionHttpClient](#) mais leve em comparação com o ApacheHttpClient padrão. O [URLConnectionHttpClient](#) é baseado no [URLConnection](#) do Java.

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

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

Note

O `URLConnectionHttpClient` não é compatível com o método HTTP PATCH. Algumas operações de API da AWS exigem solicitações PATCH. Esses nomes de operação geralmente começam com `Update*`. Veja alguns exemplos a seguir.

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

Se você puder usar o `URLConnectionHttpClient`, primeiro consulte a Referência de API para o 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` do Maven em seu arquivo `pom.xml`.

Ao contrário do `ApacheHttpClient`, o `URLConnectionHttpClient` não é adicionado automaticamente ao seu projeto; portanto, quando usado, ele deve ser declarado especificamente.

O exemplo de arquivo `pom.xml` 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>
```

Usar e configurar o `URLConnectionHttpClient`

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

Qualquer que seja a abordagem, você usa o [URLConnectionHttpClient.Builder](#) para configurar as propriedades do cliente HTTP baseado em `URLConnection`.

Prática recomendada: dedicar uma instância do `URLConnectionHttpClient` a um cliente de serviço

Se você precisar configurar uma instância do `URLConnectionHttpClient`, recomendamos que você crie a instância `URLConnectionHttpClient` dedicada. Faça isso usando o método `httpClientBuilder` do builder do cliente do 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 instância do `URLConnectionHttpClient` não for fechada quando não for mais necessária.

O exemplo a seguir cria um `S3Client` e configura a instância embutida do `URLConnectionHttpClient` com os valores `socketTimeout` e `proxyConfiguration`. O método `proxyConfiguration` usa uma expressão lambda Java do tipo `Consumer<ProxyConfiguration.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 instância do **URLConnectionHttpClient**

Para ajudar a reduzir o uso de recursos e memória do 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 instância do `URLConnectionHttpClient` 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 instância `URLConnectionHttpClient` configurada é transmitida ao método `httpClient` de cada criador. Quando os clientes do serviço e o cliente HTTP não são mais necessários, o código os fecha explicitamente. O código fecha o cliente HTTP 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();
```

Usar **URLConnectionHttpClient** e **ApacheHttpClient** em conjunto

Ao usar o `URLConnectionHttpClient` na aplicação, é necessário fornecer a cada cliente de serviço uma instância `URLConnectionHttpClient` ou `ApacheHttpClient` usando o método `httpClientBuilder` do criador do cliente de serviço.

Uma exceção ocorre se o programa usar vários clientes do serviço e estas duas condições forem verdadeiras:

- Um cliente de serviço está configurado para usar uma instância do `URLConnectionHttpClient`
- Outro cliente de serviço usa o `ApacheHttpClient` padrão sem criá-lo explicitamente com os métodos `httpClient()` ou `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 artefato `apache-client` em seu arquivo de projeto do Maven.

Exemplo de configuração do proxy

O trecho de código a seguir usa o [builder de configuração de proxy para o cliente HTTP da conexão do 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 trecho da linha de comando a seguir.

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

Note

No momento, o cliente HTTP da conexão do 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 o baseado em Netty. [NettyNioAsyncHttpClient](#) O cliente baseado em Netty é baseado na estrutura de rede assíncrona orientada por eventos do [projeto Netty](#).

Como cliente HTTP alternativo, é possível usar o novo [cliente HTTP baseado em AWS CRT](#). Este tópico mostra como configurar o `NettyNioAsyncHttpClient`.

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 configurará o `NettyNioAsyncHttpClient` com valores padrão para você.

Se você quiser configurar o `NettyNioAsyncHttpClient` explicitamente ou usá-lo com vários clientes de serviço, precisará disponibilizá-lo para configuração.

Nenhuma configuração 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 classe `NettyNioAsyncHttpClient` disponível para o código em runtime, mas não no momento da compilação. Se você não estiver configurando o cliente HTTP baseado em Netty, não precisará especificar uma dependência para ele.

No seguinte trecho XML de um arquivo `pom.xml` do Maven, a dependência declarada com `<artifactId>dynamodb-enhanced</artifactId>` traz transitivamente 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 biblioteca `NettyNioAsyncHttpClient` está apenas no caminho de classe em runtime.

Configuração necessária

Para configurar o `NettyNioAsyncHttpClient`, você precisa adicionar uma dependência no artefato `netty-nio-client` em tempo de compilação.

Consulte o exemplo a seguir de um arquivo `pom.xml` do 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>
```

Usar e configurar o `NettyNioAsyncHttpClient`

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

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

Prática recomendada: dedicar uma instância **NettyNioAsyncHttpClient** a um cliente de serviço

Se você precisar configurar uma instância do `NettyNioAsyncHttpClient`, recomendamos criar uma instância `NettyNioAsyncHttpClient` dedicada. Faça isso usando o método `httpClientBuilder` do builder do cliente do 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 instância do `NettyNioAsyncHttpClient` não for fechada quando não for mais necessária.

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

Para ajudar a reduzir o uso de recursos e memória do 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 instância `NettyNioAsyncHttpClient` 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 instância `NettyNioAsyncHttpClient` configurada é transmitida ao método `httpClient` de cada criador. Quando os clientes do serviço e o cliente HTTP não são mais necessários, o código os fecha explicitamente. O código fecha o cliente HTTP 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.builder().maxConcurrency(100).build();
```

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

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

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

DynamoDbEnhancedAsyncClient enhancedAsyncClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbClient(dbAsyncClient)

        .extensions(AutoGeneratedTimestampRecordExtension.create())
        .build();

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

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

Exemplo de configuração do proxy

O trecho de código a seguir usa o [builder 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 trecho da linha de comando a seguir.

```
$ 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 propriedade `scheme` deve ser definida no código como `https`. Se a propriedade `scheme` não estiver definida no código, o esquema usará HTTP como padrão, e o SDK procurará somente as propriedades do sistema `http.*`.

Configurar os clientes HTTP baseados em AWS CRT

Os clientes HTTP baseados em AWS CRT incluem o [AwsCrtHttpClient](#) síncrono e o [AwsCrtAsyncHttpClient](#) assíncrono. Os clientes HTTP baseados em AWS CRT oferecem os seguintes benefícios do cliente HTTP:

- Menor tempo de inicialização do SDK
- Menor espaço ocupado na memória
- Tempo de latência reduzido
- Gerenciamento de integridade da conexão
- balanceamento de carga do DNS

Componentes da AWS baseados em CRT no SDK

Os clientes HTTP baseados em AWS CRT, descritos neste tópico, e o cliente do S3 baseado em AWS CRT são componentes diferentes no SDK.

Os clientes HTTP baseados em AWS CRT síncronos e assíncronos são implementações da interface de cliente HTTP do SDK usadas para a comunicação HTTP em geral. Eles são alternativas aos outros clientes HTTP síncronos ou assíncronos no SDK, com benefícios adicionais.

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

Embora os dois componentes usem bibliotecas do [AWS Common Runtime](#), o cliente HTTP baseado em AWS CRT não usa a [biblioteca aws-c-s3](#) e não é compatível com os recursos da [API de multipart upload do S3](#). O cliente S3 da AWS baseado em CRT, por outro lado, foi criado especificamente para oferecer suporte aos recursos da API de upload em várias partes do S3.

Acessar os clientes HTTP baseados em AWS CRT

Para usar os clientes HTTP baseados em AWS CRT, adicione o artefato `aws-crt-client` com, no mínimo, a versão 2.22.0 às dependências do projeto.

O `pom.xml` Maven a seguir mostra os clientes HTTP baseados em AWSCRT declarados usando o mecanismo de lista de materiais (BOM).

```
<project>
  <properties>
    <aws.sdk.version>2.22.0</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 receber a [versão mais recente](#).

Usar e configurar um cliente HTTP baseado em AWS CRT

É possível configurar um cliente HTTP baseado em AWS CRT junto com a criação de um cliente de serviço ou configurar uma única instância para compartilhar entre vários clientes de serviço.

Com qualquer abordagem, você vai usar um criador para [configurar as propriedades](#) da instância do cliente HTTP baseado em AWS CRT.

Prática recomendada: dedicar uma instância do a um cliente de serviço

Se você precisar configurar uma instância de um cliente HTTP baseado em AWS CRT, recomendamos dedicar a instância desenvolvendo-a junto com o cliente de serviço. Faça isso usando o método `httpClientBuilder` do builder do cliente do serviço. Dessa forma, o ciclo de vida do cliente HTTP é gerenciado pelo SDK, o que ajudará a evitar possíveis vazamentos de memória se a instância do cliente HTTP baseado em AWS CRT não for fechada quando não for mais necessária.

O exemplo a seguir cria um cliente de serviço do S3 e configura um cliente HTTP baseado em AWS CRT com os valores de `connectionTimeout` e `maxConcurrency`.

Synchronous client

Importações

```
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Código

```
// Singleton: Use s3Client for all requests.
S3Client s3Client = S3Client.builder()
    .httpClientBuilder(AwsCrtHttpClient
        .builder()
        .connectionTimeout(Duration.ofSeconds(3))
        .maxConcurrency(100))
    .build();

// Perform work with the s3Client.

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

Asynchronous client

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 instância do

Para ajudar a reduzir o uso de recursos e de memória da aplicação, é possível configurar um cliente HTTP baseado em AWS CRT 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 instância do cliente HTTP baseado em AWS CRT é compartilhada, é necessário 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 baseado em AWS CRT com os valores de `connectionTimeout` e `maxConcurrency`. A instância configurada é transmitida ao método `httpClient` do criador de cada cliente de serviço. Quando os clientes do serviço e o cliente HTTP não são mais necessários, eles são explicitamente fechados. O cliente HTTP é fechado por último.

Synchronous client

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.crt.AwsCrtHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Código

```
// Create an AwsCrtHttpClient shared instance.
SdkHttpClient crtHttpClient = AwsCrtHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(3))
    .maxConcurrency(100)
    .build();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client = S3Client.builder()
    .httpClient(crtHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.crea
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();

DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .httpClient(crtHttpClient)
    .credentialsProvider(EnvironmentVariableCredentialsProvider.crea
    .defaultsMode(DefaultsMode.IN_REGION)
    .region(Region.US_EAST_1)
    .build();

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

Asynchronous client

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.builder()
    .connectionTimeout(Duration.ofSeconds(3))
    .maxConcurrency(100)
    .build();

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

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

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

Definir um cliente HTTP baseado em AWS CRT como padrão

É possível configurar o arquivo de compilação do Maven para que o SDK use um cliente HTTP baseado em AWS CRT como o cliente HTTP padrão para clientes de serviço.

Você faz isso adicionando um elemento `exclusions` com as dependências padrão do cliente HTTP a cada artefato do cliente de serviço.

No exemplo `pom.xml` a seguir, o SDK usa um cliente HTTP baseado em AWS CRT para serviços do S3. Se o cliente de serviço no código for um `S3AsyncClient`, o SDK usará `AwsCrtAsyncHttpClient`. Se o cliente de serviço for um `S3Client`, o SDK usará `AwsCrtHttpClient`. Com essa configuração, o cliente HTTP assíncrono padrão baseado em Netty e o HTTP síncrono padrão baseado em Apache não estão disponíveis.

```
<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>
        <exclusion>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>apache-client</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
    <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 de [VERSION](#).

Note

Se vários clientes de serviço forem declarados em um arquivo `pom.xml`, todos precisarão do elemento XML `exclusions`.

Usar uma propriedade do sistema Java

Para usar os clientes HTTP baseados em AWS CRT como o HTTP padrão para a aplicação, é possível definir a propriedade `software.amazon.awssdk.http.async.service.impl` do sistema Java com um valor de `software.amazon.awssdk.http.crt.AwsCrtSdkHttpClient`.

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.AwsCrtSdkHttpClient
```

Use o trecho de código a seguir 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.AwsCrtSdkHttpClient");
```

Note

É necessário adicionar uma dependência do artefato `aws-crt-client` no arquivo `pom.xml` ao usar uma propriedade do sistema para configurar o uso do cliente HTTP baseado em AWS CRT.

Configuração avançada de clientes HTTP baseados em AWS CRT

É possível usar várias configurações dos clientes HTTP baseados em AWS CRT, incluindo a configuração de integridade da conexão e o tempo máximo de inatividade. Você pode revisar as [opções de configuração disponíveis](#) para o `AwsCrtAsyncHttpClient`. É possível configurar as mesmas opções para o `AwsCrtHttpClient`.

Configuração de integridade da conexão

É possível configurar a integridade da conexão para o cliente HTTP baseado em AWS CRT usando o método `connectionHealthConfiguration` no criador de clientes HTTP.

O exemplo a seguir cria um serviço do S3 que usa uma instância do cliente HTTP baseado em AWS CRT definida com a configuração de integridade da conexão e um tempo máximo de inatividade para as conexões.

Synchronous client

Importações

```
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

Código

```
// Singleton: Use the s3Client for all requests.
S3Client s3Client = S3Client.builder()
    .httpClientBuilder(AwsCrtHttpClient
        .builder()
        .connectionHealthConfiguration(builder -> builder
            .minimumThroughputInBps(32000L)
            .minimumThroughputTimeout(Duration.ofSeconds(3)))
        .connectionMaxIdleTime(Duration.ofSeconds(5)))
    .build();

// Perform work with s3Client.

// Requests complete: Close the service client.
s3Client.close();
```

Asynchronous client

Importações

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

Código

```
// 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 aceito nos clientes HTTP baseados em AWS CRT, mas está planejado para uma versão futura.

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

Exemplo de configuração do proxy

O trecho de código a seguir mostra o uso do [ProxyConfiguration.Builder](#) que você usa para definir a configuração de proxy no código.

Synchronous client

Importações

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtHttpClient;
import software.amazon.awssdk.http.crt.ProxyConfiguration;
```

Código

```
SdkHttpClient crtHttpClient = AwsCrtHttpClient.builder()
    .proxyConfiguration(ProxyConfiguration.builder()
        .scheme("https")
        .host("myproxy")
        .port(1234)
        .username("username")
        .password("password")
        .build())
    .build();
```

Asynchronous client

Importações

```
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.crt.AwsCrtAsyncHttpClient;
import software.amazon.awssdk.http.crt.ProxyConfiguration;
```

Código

```
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 trecho da linha de comando a seguir.

```
$ 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 propriedade `scheme` deve ser definida no código como `https`. Se a propriedade `scheme` não estiver definida no

código, o esquema usará HTTP como padrão, e o SDK procurará somente as propriedades do sistema `http.*`.

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 desmarcadas?

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 instância do Amazon EC2 que não existe, o Amazon EC2 retornará uma resposta de erro, e todos os detalhes dessa resposta de erro serão incluídos na [AwsServiceException](#) lançada.

Ao encontrar um [AwsServiceException](#), você sabe que a requisição foi enviada com êxito para o AWS service (Serviço da AWS), mas não foi possível processá-la com êxito. 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 erro da AWS retornado
- Mensagem de erro detalhada do serviço na [AwsErrorDetails](#) classe
- ID de requisição da AWS para a requisição com falha

Para alguns casos, uma subclasse de `AwsServiceException` específica do serviço é lançada para permitir que os desenvolvedores tenham o controle refinado do tratamento dos casos de erro nos blocos catch. 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 Serviços da AWS comuns. 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 valor do `errorCode` para pesquisar informações na API do guia de serviços. Por exemplo, se uma `S3Exception` for detectada e o valor de `AwsErrorDetails#errorCode()` for `InvalidRequest`, 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 `SdkClientException` normalmente é mais grave do que um `SdkServiceException` e indica um problema grave que esteja evitando que o cliente faça chamadas de serviço para serviços da AWS. 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 nova tentativa

O SDK for Java faz novas tentativas de solicitações para várias [exceções do lado do cliente](#) e para [códigos de status HTTP](#) que ele recebe das respostas do AWS service (Serviço da AWS). Esses

erros são tratados como parte do `RetryMode` legado 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 uma [RetryPolicy](#) que adicione instâncias [RetryOnExceptionsCondition](#) e [RetryOnStatusCodeCondition](#).

Registro em log com o SDK para Java 2.x

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

Entre os sistemas de registro em log compatíveis estão 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, chamado `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 arquivo `log4j2.xml` precisa estar no caminho de classe quando seu aplicativo é inicializado. Para um projeto do Maven, coloque o arquivo no diretório `<project-dir>/src/main/resources`.

O arquivo de configuração `log4j2.xml` especifica propriedades como [nível de registro em log](#), em que a saída do registro em log 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 dá suporte ao conceito de múltiplas [hierarquias](#) de registro em log. 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 vinculação do Log4j 2 para SLF4J no arquivo de compilação, use o seguinte.

Maven

Adicione os elementos a seguir ao arquivo `pom.xml`:

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

Gradle–Kotlin DSL

Adicione o seguinte ao arquivo `build.gradle.kts`:

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

A versão mínima do artefato `log4j-slf4j2-impl` a ser usada é a `2.20.0`. Para a versão mais recente, use a versão publicada no [Maven central](#). Substitua **VERSÃO** pela versão que você usará.

Erros e avisos específicos do SDK

Recomendamos sempre deixar a hierarquia do registrador em log `"software.amazon.awssdk"` definida como `"WARN"` para interceptar todas as mensagens importantes das bibliotecas de cliente do SDK. Por exemplo, se o cliente do Amazon S3 detectar que o aplicativo não fechou corretamente um `InputStream` e possa estar vazando recursos, o cliente do S3 informará isso por meio de uma mensagem de aviso para os logs. Isso também garante que as mensagens serão registradas em log se o cliente enfrentar algum problema ao processar requisições ou respostas.

O arquivo `log4j2.xml` a seguir define o `rootLogger` como `"AVISO"`, o que faz com que mensagens de aviso e nível de erro de todos os registradores do aplicativo sejam enviadas, incluindo aqueles na hierarquia `"software.amazon.awssdk"`. Você também pode definir explicitamente

a hierarquia do registrador em log “software.amazon.awssdk” como “AVISO” se `<Root level="ERROR">` for usado.

Exemplo de arquivo de configuração Log4j2.xml

Essa configuração registrará mensagens nos níveis “ERRO” e “AVISO” 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 em log do resumo de requisição/resposta

Cada solicitação para um AWS service (Serviço da AWS) gera um ID de solicitação exclusivo da AWS que é útil se você tiver problemas com a forma como um AWS service (Serviço da AWS) está lidando com uma solicitação. Os IDs de solicitação da AWS podem ser acessados programaticamente por meio de objetos de [SdkServiceException](#) 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 arquivo `log4j2.xml` a seguir permite um resumo de 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:
DefaultSdkHttpRequestFullRequest(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 de solicitação, use `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`.

Registro em log detalhado

Em alguns casos, pode ser útil ver as requisições e as respostas exatas enviadas e recebidas pelo SDK para Java 2.x. 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 detalhamento dos clientes HTTP que podem ser usados para as duas categorias de clientes de serviço:

Cientes síncronos	Cientes de HTTP assíncronos
ApacheHttpClient (padrão)	NettyNioAsyncHttpClient (padrão)
URLConnectionHttpClient	AwsCrtAsyncHttpClient

Consulte a guia apropriada abaixo para ver as configurações que você precisa adicionar conforme o 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 configuração `log4j2.xml` 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>
```

É necessária uma dependência adicional do Maven no artefato `log4j-1.2-api` para o registro de transmissão de dados com o Apache, pois ele usa 1.2 internamente.

O conjunto completo de dependências do Maven para o log4j 2, incluindo o registro de transmissão de dados 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")
}
```

...

A versão mínima do artefato `log4j-bom` a ser usada é a `2.20.0`. Para a versão mais recente, use a versão publicada no [Maven central](#). Substitua **VERSÃO** pela versão que você usará.

URLConnectionHttpClient

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

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

Essa configuração registrará somente os cabeçalhos de solicitação e de 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=e367fa0bc217a6a65675bb743e1280cf12f8e8d566196a816d948fdf0b42ca1a}{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: sAFeZD0kDUMsBbkdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
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 de solicitação e de 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 "POST" para acessar rapidamente a seção de log que contém solicitações e respostas reais. Pesquise "Plaintext before ENCRYPTION" para solicitações e "Plaintext after DECRYPTION" para respostas para ver o texto completo dos cabeçalhos e corpos.

NettyNioAsyncHttpClient

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

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

Veja aqui um exemplo de `log4j2.xml` 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 de cabeçalho e corpos de solicitação e resposta.

AwsCrtAsyncHttpClient

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

Log to a file at "Debug" level

Como usar 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

Como usar 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 "Rastreamento" o `AwsCrtAsyncHttpClient` registra somente cabeçalhos de resposta. Cabeçalhos de solicitação, corpos de solicitação e corpos de resposta não são registrados.

Definir o JVM TTL para pesquisas de nome 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 (TTL). time-to-live

Como AWS os recursos usam entradas de nome DNS que mudam ocasionalmente, recomendamos que você configure sua JVM com um valor TTL de no máximo 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. Portanto, se o endereço IP de um AWS recurso mudar enquanto seu aplicativo ainda estiver em execução, ele não poderá usar esse recurso até que você reinicie manualmente a JVM e as informações IP em cache sejam 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 definir 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");
```

Práticas recomendadas do AWS SDK for Java 2.x

Esta seção lista as melhores práticas para usar o SDK para Java 2.x.

Tópicos

- [Reutilizar um cliente do SDK, se possível](#)
- [Fechar os streams de entrada das operações do cliente](#)
- [Ajustar as configurações HTTP com base em testes de desempenho](#)
- [Usar o OpenSSL para o cliente HTTP baseado no Netty](#)
- [Configurar tempos limite da API](#)
- [Usar métricas](#)

Reutilizar um cliente do SDK, se possível

Cada cliente do SDK mantém seu próprio grupo de conexões HTTP. Uma conexão que já existe no grupo 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 grupos de conexão que não são usados de forma eficaz. Todos os clientes do SDK são seguros para os threads.

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

Fechar os streams de entrada das operações do cliente

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

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

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

Ajustar 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 para seus aplicativos com base em seus casos de uso.

Como um bom ponto de partida, o SDK oferece um atributo [de padrões de configuração inteligentes](#). Esse atributo está disponível somente na versão 2.17.102. Escolha um modo de acordo com seu caso de uso que forneça valores de configuração sensatos.

Usar o OpenSSL para o cliente HTTP baseado no Netty

Por padrão, o [NettyNioAsyncHttpClient](#) do SDK usa a implementação SSL padrão do JDK como o `SslProvider`. Nossos testes descobriram que o OpenSSL tem um desempenho melhor do que a implementação padrão do JDK. A comunidade do 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 configurar `netty-tcnative` para seu projeto, a instância `NettyNioAsyncHttpClient` selecionará automaticamente o OpenSSL. Você também pode definir o `SslProvider` explicitamente usando o construtor `NettyNioAsyncHttpClient`, conforme mostrado no trecho a seguir.

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

Configurar tempos limite da API

O SDK fornece [valores padrão](#) para algumas opções de tempos 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 chamadas de API individuais. É uma boa prática definir tempos limite para as tentativas individuais e para toda a solicitação. Isso garantirá que seu aplicativo se antecipe à falha de maneira ideal quando houver problemas transitórios que podem fazer com que as tentativas de solicitação demorem mais para serem concluídas ou problemas fatais na rede.

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

O exemplo a seguir mostra a configuração de um cliente do 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 o qual a chamada de API pode ser repetida.

apiCallTimeout

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

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

O exemplo a seguir configura uma única solicitação `listBuckets` 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 em conjunto, você define um limite rígido para o tempo total gasto em todas as tentativas das repetições. Você também configura uma solicitação HTTP individual para se antecipar à falha rapidamente em uma solicitação lenta.

Usar métricas

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

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

Use os recursos do AWS SDK for Java 2.x

Recursos gerais

O SDK para Java 2.x contém vários recursos que facilitam a programação nos Serviços da AWS .

- O SDK oculta os mecanismos complexos por trás da [recuperação de resultados paginados](#) e da [pesquisa de recursos](#).
- A [programação assíncrona com E/S sem bloqueio](#) ajuda você a escrever código simultâneo com melhor desempenho. O SDK oferece os benefícios do [HTTP/2](#), por exemplo, latência reduzida, sempre que possível.
- O SDK do Java pode gerar [métricas](#) para ajudar a monitorar a integridade operacional das aplicações.

Recursos específicos do serviço

Além dos recursos gerais mencionados anteriormente, o Java SDK fornece recursos específicos Serviços da AWS.

- Amazon S3: para [simplificar o trabalho com arquivos e diretórios](#) com o Amazon S3, o SDK oferece o S3 Transfer Manager. Para [melhorar o desempenho e a confiabilidade](#) ao usar a API S3 assíncrona padrão do SDK, o SDK oferece o cliente S3 baseado em CRT. AWS
- DynamoDB: [o recurso de mapeamento orientado a objetos](#) é oferecido pela API do cliente aprimorado do DynamoDB. [Trabalhe com dados orientados a documentos no estilo JSON](#) usando a API de documentos aprimorada.
- IAM: a API do IAM Policy Builder oferece uma [forma segura e orientada a objetos de criar políticas do IAM](#).

Trabalhar com resultados paginados usando o AWS SDK for Java 2.x

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 do Amazon S3 .

Iterar sobre páginas

O primeiro exemplo demonstra o uso de um objeto `listRes` paginador, uma [ListObjectsV2Iterable](#) instância, 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 [S3Object](#) conteúdo e, em seguida, processa o conteúdo do Amazon S3 objeto.

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.

Iterar sobre objetos

Os exemplos a seguir mostram maneiras de iterar sobre os objetos retornados na resposta e não nas páginas de resposta. O método `contents` da classe `ListObjectsV2Iterable` retorna um [SdkIterable](#) que fornece vários métodos para processar os elementos do conteúdo subjacente.

Usar um stream

O seguinte trecho usa o método `stream` no conteúdo de 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.

Usar um loop for-each

Como `SdkIterable` estende a interface `Iterable`, você pode processar o conteúdo como qualquer outro `Iterable`. O trecho a seguir usa um loop `for-each` 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 o loop `while`.

```
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

Iterar 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.

Usar um **Subscriber**

O exemplo de código a seguir demonstra como processar resultados paginados usando a interface `org.reactivestreams.Publisher` 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.paginator.ListTablesPublisher;

import java.util.List;
```

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

O código a seguir adquire uma instância `ListTablesPublisher`.

```
// 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 método `onNext` 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 publicador. Este 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.

Usar um **Consumer**

A interface do `SdkPublisher` que `ListTablesPublisher` implementa tem um método `subscribe` que pega um `Consumer` e retorna um `CompletableFuture<Void>`.

O método `subscribe` dessa interface pode ser usado para casos de uso simples, quando um `org.reactivestreams.Subscriber` pode ser uma sobrecarga. Como o código abaixo consome cada página, ele chama o método `tableNames` em cada uma. O método `tableNames` 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.

Iterar sobre nomes de tabela

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 seu método `contents`, a classe de resultados assíncronos do DynamoDB, `ListTablesPublisher` tem o método conveniente `tableNames` para interagir com a coleção de itens subjacente. O tipo de retorno do método `tableNames` é um [SdkPublisher](#) que pode ser usado para solicitar itens em todas as páginas.

Usar um **Subscriber**

O código a seguir adquire um `SdkPublisher` 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 método `onNext` 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 publicador. Este método é chamado repetidamente até que todos os nomes de 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.

Usar um **Consumer**

O exemplo a seguir usa o método `subscribe` do `SdkPublisher` que utiliza um `Consumer` para processar cada item.

```
// Use a Consumer.
CompletableFuture<Void> future = publisher.subscribe(System.out::println);
future.get();
```

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

Usar bibliotecas 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::tableName)
    .toList()
    .blockingGet();
System.out.println(tables);
```

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

Pesquisa de estados de recursos no AWS SDK for Java 2.x:

Garçons

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 projeto (por exemplo, no seu arquivo `pom.xml` ou `build.gradle`) 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>
```

Usar agentes de espera

Para instanciar um objeto de waiters, primeiro crie um cliente de serviço. Defina o método `waiter()` do cliente de serviço como o valor do objeto `waiter`. 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 waiters

Você pode personalizar a configuração de um waiter usando o `overrideConfiguration()` em seu construtor. Para algumas operações, você pode aplicar uma configuração personalizada ao fazer a solicitação.

Configurar um waiter

O trecho de código a seguir mostra como substituir a configuração em um waiter.

```

// sync
DynamoDbWaiter waiter =
    DynamoDbWaiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(10))
        .client(dynamoDbClient)
        .build();

// async
DynamoDbAsyncWaiter asyncWaiter =
    DynamoDbAsyncWaiter.builder()
        .client(dynamoDbAsyncClient)
        .overrideConfiguration(o -> o.backoffStrategy(
            FixedDelayBackoffStrategy.create(Duration.ofSeconds(2))))
        .scheduledExecutorService(Executors.newScheduledThreadPool(3))
        .build();

```

Substituir configuração para uma solicitação específica

O trecho de código a seguir mostra como substituir a configuração de um waiter por solicitação. Observe que somente algumas operações têm configurações personalizáveis.

```
waiter.waitForTableNotExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));

asyncWaiter.waitForTableExists(b -> b.tableName("myTable"),
    o -> o.waitFor(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

Usar programação assíncrona

Ele AWS SDK for Java 2.x apresenta clientes assíncronos com suporte de E/S sem bloqueio que implementam alta simultaneidade em alguns segmentos. No entanto, a E/S total sem bloqueio não é garantida. O cliente assíncrono pode realizar chamadas de bloqueio em alguns casos, como recuperação de credenciais, assinatura de solicitações usando [AWS Signature Version 4 \(SigV4\)](#) ou descoberta de endpoints.

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

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. 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 desejado. Normalmente, é assim que você passa os 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 carrega um arquivo de forma Amazon S3 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 de forma Amazon S3 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 `ExecutorService` por trás do Netty para concluir os futuros retornos da solicitação do cliente HTTP para o cliente Netty. Essa abstração reduz o risco de um aplicativo interromper o processo assíncrono se os desenvolvedores optarem por suspender ou desabilitar 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();
```

Trabalhe com HTTP/2 no AWS SDK for Java

HTTP/2 é uma revisão importante 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 que atualizarem 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 [Como trabalhar com o Kinesis](#).

Use as métricas do SDK do 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 a habilitar e configurá-lo.

Pré-requisitos

Antes de habilitar e usar métricas, é necessário seguir essas etapas:

- Siga as etapas em [Configuração](#).
- Configure as dependências do seu projeto (por exemplo, no seu arquivo `pom.xml` ou `build.gradle`) 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>
```

- Habilite as permissões `cloudwatch:PutMetricData` para a identidade do IAM para permitir que o SDK para Java escreva métricas.

Como ativar a coleta de métricas

Você pode habilitar métricas em seu aplicativo para um cliente de serviço ou em solicitações individuais.

Habilitar 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 publicador de métricas.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
DynamoDbClient ddb = DynamoDbClient.create();
ddb.listTables(ListTablesRequest.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build());
```

Habilitar 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 publicador 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();
```

Quando as métricas estão disponíveis?

As métricas geralmente ficam disponíveis em 5 a 10 minutos após a emissão das métricas pelo SDK para Java. Para obter up-to-date métricas precisas, verifique o Cloudwatch pelo menos 10 minutos depois de emitir as métricas de seus aplicativos Java.

Quais informações são coletadas?

A coleção de métricas inclui o seguinte:

- Número de solicitações de API, incluindo se foram bem-sucedidas ou se falharam
- Informações sobre os AWS serviços que você chama em suas solicitações de API, incluindo exceções retornadas
- A duração de várias operações, como marshalling, 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 de serviço do cliente](#).

Como posso usar essas informações?

Use as métricas que o SDK coleta para monitorar os clientes de serviço em sua aplicação. Você pode analisar as tendências gerais de uso, identificar anomalias, analisar as exceções retornadas pelos clientes de serviço ou obter mais informações 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 de 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 é possível coletar e os requisitos de uso do cliente HTTP.

Para obter mais informações sobre como habilitar e configurar métricas para o SDK, consulte [Como habilitar métricas do SDK](#).

Os termos usados nas tabelas significam:

- Apache: o cliente HTTP baseado no Apache ([ApacheHttpClient](#))
- Netty: o cliente HTTP baseado em Netty ([NettyNioAsyncHttpClient](#))
- CRT: o cliente HTTP AWS baseado em CRT () [AwsCrtAsyncHttpClient](#)
- Todos: a coleta de dados métricos não depende do cliente HTTP; isso inclui o uso do cliente HTTP baseado em conexão URL ([URLConnectionHttpClient](#))

Métricas coletadas com cada solicitação

Nome da métrica	Descrição	Tipo	Um cliente HTTP é necessário
ApiCallDuration	O tempo total necessário para concluir uma solicitação	Duração	Any

Nome da métrica	Descrição	Tipo	Um cliente HTTP é necessário
	ão (incluindo todas as novas tentativas)		
ApiCallSuccessful	Verdadeiro se a chamada de API foi bem-sucedida; falso se não	Booleano	Any
CredentialsFetchDuration	O tempo necessário para obter as credenciais de AWS assinatura da solicitação	Duração	Any
MarshallingDuration	O tempo necessário para organizar a solicitação	Duração	Any
OperationName	O nome da AWS API para a qual a solicitação é feita	String	Any
RetryCount	Número de vezes que o SDK tentou novamente realizar a chamada de API	Inteiro	Any
ServiceId	ID de serviço do AWS service (Serviço da AWS) qual a solicitação de API foi feita	String	Any

Nome da métrica	Descrição	Tipo	Um cliente HTTP é necessário
TokenFetchDuration	O tempo necessário para buscar as credenciais de assinatura do token para a solicitação	Duração	Any

Métricas coletadas para cada tentativa de solicitação

Cada chamada de API pode exigir várias tentativas antes de receber uma resposta. Essas métricas são coletadas para cada tentativa.

Nome da métrica	Descrição	Tipo	Um cliente HTTP é necessário
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	A ID da solicitação estendida da solicitação de serviço	String	Any
AwsRequestId	A ID da solicitação de serviço	String	Any
BackoffDelayDuration	O tempo que o SDK esperou antes dessa tentativa de chamada de API	Duração	Any

Nome da métrica	Descrição	Tipo	Um cliente HTTP é necessário
ConcurrencyAcquireDuration	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	String	Apache, Netty, CRT
HttpStatusCode	O código de status retornado com a resposta HTTP	Inteiro	Any
LeasedConcurrency	O número de solicitações que estão sendo executadas atualmente e 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	Any
MaxConcurrency	O número máximo de solicitações simultâneas suportadas pelo cliente HTTP	Inteiro	Apache, Netty, CRT

Nome da métrica	Descrição	Tipo	Um cliente HTTP é necessário
PendingConcurrencyAcquires	O número de solicitações que estão bloqueadas, aguardando outra conexão TCP ou a disponibilidade 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	Any
SigningDuration	O tempo necessário para assinar a solicitação HTTP	Duração	Any
UnmarshallingDuration	O tempo necessário para desvincular uma resposta HTTP de uma resposta do SDK	Duração	Any

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 o select. Serviços da AWS Para ver um conjunto completo de exemplos, consulte a [seção Exemplos de código](#).

Tópicos

- [Trabalhar com CloudWatch](#)
- [serviços de banco de dados AWS e AWS SDK for Java 2.x](#)
- [Trabalhar com DynamoDB](#)
- [Trabalhe com Amazon EC2](#)
- [Trabalhar com IAM](#)
- [Trabalhe com Kinesis](#)
- [Invocar, listar e excluir funções do AWS Lambda](#)
- [Trabalhar com o Amazon S3](#)
- [Trabalhar com Amazon Simple Notification Service](#)
- [Trabalhe com Amazon Simple Queue Service](#)
- [Trabalhar com Amazon Transcribe](#)

Trabalhar com CloudWatch

Esta seção fornece exemplos de como programar o [Amazon 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.

Os exemplos a seguir incluem somente o código necessário para demonstrar cada técnica. O [código de exemplo completo está disponível no 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](#)
- [Publicar dados de métrica personalizada no CloudWatch](#)
- [Trabalho com alarmes do CloudWatch](#)
- [Use os CloudWatch eventos da Amazon](#)

Obtenha métricas de CloudWatch

Listar métricas

Para listar CloudWatch as métricas, crie um `listMetrics` método [ListMetricsRequeste](#) chame o `CloudWatchClient`. 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 por serviços da AWS pode ser encontrada na [Referência de métricas e dimensões do Amazon CloudWatch](#) no Guia do usuário Amazon CloudWatch.

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 a [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

Publicar dados de métrica personalizada no CloudWatch

Vários serviços da AWS publicam [as próprias métricas](#) em namespaces que começam com “AWS”. Também é possível publicar dados de métricas personalizadas usando seu próprio namespace (contanto 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 começa com “AWS”. Namespaces que começam com “AWS” são reservados para serem usados por produtos da Amazon Web Services.

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](#) em GitHub.

Mais informações

- [Use Amazon CloudWatch métricas](#) no Guia do Amazon CloudWatch usuário.
- [Namespaces da AWS](#) no Guia do usuário Amazon CloudWatch.
- [PutMetricData](#) na Referência da Amazon CloudWatch API.

Trabalho com alarmes do CloudWatch

Criar um alarme

Para criar um alarme com base em uma CloudWatch métrica, chame o `putMetricAlarm` método `CloudWatchClient`'s [PutMetricAlarmRequest](#) preenchendo 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 pode 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 desCWAAlarms( 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 `CloudWatchClient's describeAlarmsForMetric`. O uso é semelhante a `describeAlarms`.

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

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 deleteCWAlarm(CloudWatchClient cw, String alarmName) {

    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

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

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

Mais informações

- [Usando Amazon CloudWatch 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 os CloudWatch eventos da Amazon

CloudWatch O Events fornece 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 destinos integrados. Você pode comparar eventos e roteá-los para um ou mais fluxos ou funções de destino usando regras simples.

A Amazon EventBridge é a [evolução](#) dos CloudWatch eventos. Ambos os serviços usam a mesma API, para que você possa continuar usando o [cliente de CloudWatch eventos](#) fornecido pelo SDK ou migrar para o [EventBridge cliente](#) do SDK for Java para a funcionalidade de eventos. CloudWatch CloudWatch A [documentação do Guia do Usuário](#) de Eventos e a [referência da API](#) agora estão disponíveis nos sites de EventBridge documentação.

Adicionar eventos

Para adicionar CloudWatch eventos personalizados, chame o `CloudWatchEventsClient`'s `putEvents` método 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 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);
}
}
```

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

Adicionar regras

Para criar ou atualizar uma regra, chame o `CloudWatchEventsClient`'s `putRule` método 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",
            ruleArn, 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. Exemplos de alvos incluem Amazon EC2 instâncias, Lambda funções, Kinesis fluxos, Amazon ECS tarefas, máquinas de Step Functions estado e destinos integrados.

Para adicionar um destino a uma regra, chame o `CloudWatchEventsClient`'s `putTargets` método [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 putCWTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        PutTargetsResponse response = cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

Mais informações

- [Adicionando eventos PutEvents](#) no Guia do EventBridge usuário da Amazon
- [Agende expressões para regras](#) no Guia do EventBridge usuário da Amazon
- [Tipos de eventos para CloudWatch Events](#) o Amazon EventBridge User Guide
- [Padrões de eventos](#) no Guia EventBridge do usuário da Amazon
- [PutEvents](#) na Amazon EventBridge API Reference
- [PutTargets](#) na Amazon EventBridge API Reference
- [PutRule](#) na Amazon EventBridge API Reference

serviços de banco de dados AWS e AWS SDK for Java 2.x

O AWS oferece vários tipos de banco de dados: relacional, chave-valor, na 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 no AWS.

Alguns serviços de banco de dados, por exemplo, o serviço [Amazon DynamoDB](#), têm APIs de serviços web para gerenciar o recurso AWS (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 serviços web para trabalhar com os dados. O SDK for Java 2.x tem uma interface [DocDbClient](#) correspondente para trabalhar com o recurso. No entanto, você precisa de outra API Java, como [MongoDB para Java](#), para trabalhar com os dados.

Use os exemplos abaixo para saber como usar o SDK para clientes de serviço Java 2.x com os diferentes tipos de bancos de dados.

Exemplos do Amazon DynamoDB

Trabalhar com os dados

SDK service client: [DynamoDBClient](#)

Example: [Aplicativo React/Spring REST usando o DynamoDB](#)

Examples: [Vários exemplos do DynamoDB](#)

SDK service client: [DynamoDB EnhancedClient](#)

Example: [Aplicativo React/Spring REST usando o DynamoDB](#)

Examples: [Vários exemplos do DynamoDB](#) (names starting with 'Enhanced')

Trabalhar com o banco de dados

SDK service client: [DynamoDBClient](#)

Examples: [CreateTable, ListTables, DeleteTable](#)

Veja [exemplos adicionais do DynamoDB](#) na seção de exemplos de código guiado deste guia.

Exemplos do Amazon RDS

Trabalhar com os dados	Trabalhar com o banco de dados
API não SDK: JDBC, sabor SQL específico do banco de dados; seu código gerencia conexões de banco de dados ou um pool de conexões.	Cliente de serviço SDK: RdsClient
Exemplo: aplicativo React/Spring REST usando MySQL	Exemplos: vários RdsClient exemplos

Exemplos do Amazon Redshift

Trabalhar com os dados	Trabalhar com o banco de dados
Cliente de serviço SDK: RedshiftDataClient	Cliente de serviço SDK: RedshiftClient
Exemplos: vários RedshiftDataClient exemplos	Exemplos: vários RedshiftClient exemplos
Exemplo: aplicativo React/Spring REST usando RedshiftDataClient	

Exemplos do Amazon Aurora Sem Servidor v1

Trabalhar com os dados	Trabalhar com o banco de dados
Cliente de serviço SDK: RdsDataClient	Cliente de serviço SDK: RdsClient
Exemplo: aplicativo React/Spring REST usando RdsDataClient	Exemplos: vários RdsClient exemplos

Exemplos do Amazon DocumentDB

Trabalhar com os dados	Trabalhar com o banco 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 padrão do DynamoDB de baixo nível ([DynamoDbClient](#)) oferecido com o AWS SDK for Java 2.x.

O SDK também oferece o [Cliente aprimorado do DynamoDB](#), que fornece uma abordagem de alto nível orientada a objetos para trabalhar com o DynamoDB.

Tópicos

- [Trabalhe com tabelas em DynamoDB](#)
- [Trabalhar com itens no DynamoDB](#)
- [Associar objetos Java a itens do DynamoDB com o AWS SDK for Java 2.x](#)

Trabalhe com tabelas em DynamoDB

As tabelas são os contêineres para todos os itens em um DynamoDB banco de dados. Antes de adicionar ou remover dados de DynamoDB, você deve criar uma tabela.

Para cada tabela, você deve definir:

- Um nome de tabela exclusivo para sua conta e 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 de 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 somente a capacidade que você acha que precisará para sua mesa.

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 `DynamoDbClient`'s `createTable` método para criar uma nova DynamoDB tabela. 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](#) uma 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](#) objeto 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. Os dois 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 região específica chamando o `DynamoDbClient`'s `listTables` método.

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 método `DynamoDbClient`'s `describeTable` 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](#) em GitHub.

Modificar (atualizar) uma tabela

Você pode modificar os valores de throughput provisionada da 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 método `DynamoDbClient`'s `deleteTable` 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 desenvolvedor do Amazon DynamoDB
- [Trabalhando com tabelas DynamoDB no](#) Guia do Amazon DynamoDB desenvolvedor

Trabalhar com itens no 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 desenvolvedor do Amazon DynamoDB.

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](#) em GitHub.

Recuperar (obter) 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 desejado.

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](#) em GitHub.

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 passou não existir, 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.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 `deleteItem` método `DynamoDbClient`'s 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 passou não existir, um [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](#) em GitHub.

Mais informações

- [Diretrizes para trabalhar com itens](#) no Guia do desenvolvedor do Amazon DynamoDB
- [Trabalho com itens no DynamoDB](#) no Guia do Desenvolvedor do Amazon DynamoDB

Associar objetos Java a itens do DynamoDB com o AWS SDK for Java 2.x

A [API do Cliente Aprimorado do DynamoDB](#) é uma biblioteca de alto nível que é a sucessora da classe `DynamoDBMapper` no SDK para Java v1.x. Ela oferece uma maneira simples de mapear

classes do lado do cliente para tabelas do DynamoDB. Basta definir 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 Cliente Aprimorado do DynamoDB também inclui a [API de Documento Aprimorado](#), que permite trabalhar com itens do tipo documento que não seguem um esquema definido.

A API do Cliente Aprimorado do DynamoDB será abordada nos tópicos a seguir.

- [Conceitos básicos da API do Cliente Aprimorado do DynamoDB](#)
- [Saiba mais sobre os fundamentos da API do cliente avançado do DynamoDB](#)
- [Usar recursos avançados de mapeamento](#)
- [Trabalhar com documentos JSON com a API de documentos aprimorados do DynamoDB](#)
- [Usar extensões](#)
- [Usar a API do cliente avançado do DynamoDB de forma assíncrona](#)
- [Anotações de classes de dados](#)

Conceitos básicos da API do Cliente Aprimorado do DynamoDB

O tutorial a seguir apresenta os fundamentos necessários para trabalhar com a API do Cliente Aprimorado do DynamoDB.

Adicionar dependências

Para começar a trabalhar com a API do Cliente Aprimorado do DynamoDB em seu projeto, adicione uma dependência no artefato `dynamodb-enhanced` do Maven. Essa ação será mostrada 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 substitua a `<VERSION>` por este 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 substitua a `<VERSION>` por este valor.

Gere um **TableSchema** a partir de uma classe de dados

O [TableSchema](#) permite que o cliente avançado mapeie valores de atributos do DynamoDB de e para suas classes do lado do cliente. Neste tutorial, você aprenderá sobre TableSchemas derivados de uma classe de dados estática e gerados a partir de código usando um construtor.

Usar uma classe de dados anotada

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

tenha getters e setters para cada atributo na classe. Inclua uma anotação de classe para indicar que a classe de dados é uma `DynamoDbBean`. Além disso, no mínimo, inclua uma anotação `DynamoDbPartitionKey` no getter ou setter do atributo da chave primária.

Você pode aplicar [anotações em nível de atributo](#) a getters ou setters, mas não a ambos.

Note

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

A classe `Customer` 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 o `TableSchema`, como será mostrado no trecho a seguir.

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

Um `TableSchema` é 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 classe `Person` na seção [???](#).

Usar 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 classe `Customer` 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();

```

Criar um cliente aprimorado e uma **DynamoDbTable**

Criar 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 [DynamoDbClient](#) padrão para realizar operações. A API oferece duas maneiras de criar uma instância [DynamoDbEnhancedClient](#). A primeira opção, mostrada no trecho a seguir, cria um [DynamoDbClient](#) padrão com configurações padrão retiradas das definições de configuração.

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

Se quiser configurar o cliente padrão subjacente, você poderá 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();
```

Criar uma instância **DynamoDbTable**

Pense em uma [DynamoDbTable](#) como a representação do lado do cliente de uma tabela do DynamoDB que usa a funcionalidade de mapeamento fornecida por um `TableSchema`. A classe `DynamoDbTable` 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 fábrica `table()` do `DynamoDbEnhancedClient` para criar uma instância `DynamoDbTable` conforme será mostrado no trecho a seguir.

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

Instâncias `DynamoDbTable` 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 do DynamoDB pode ou não existir. Se a tabela chamada `Customer` já existir, você poderá começar a realizar operações CRUD nela. Se ela não existir, use a instância `DynamoDbTable` para criar a tabela conforme será discutido na próxima seção.

Criar uma tabela do DynamoDB se for necessário

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

Exemplo de código para criar tabela

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

Este exemplo cria uma tabela do DynamoDB com o nome `Customer`, que é idêntico ao nome da classe, mas o nome da tabela pode ser diferente. Seja qual for o nome da tabela, você deverá usar

esse nome em aplicativos adicionais para trabalhar com a tabela. Forneça esse nome ao método `table()` sempre que criar outro objeto de `DynamoDbTable` para trabalhar com a tabela subjacente do DynamoDB.

O parâmetro Java `builder`, passado para o método `createTable`, permite que você [personalize a tabela](#). Neste exemplo, o [throughput provisionado](#) está configurado. 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 o throughput provisionado não são definidos. Em vez disso, o modo de faturamento 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 waiter significa que você não precisa escrever uma lógica que pesquise no 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
        .waitUntilTableExists(builder -> builder.tableName("Customer").build())
        .matched();
    DescribeTableResponse tableDescription = response.response().orElseThrow(
        () -> new RuntimeException("Customer table was not created."));
    // The actual error can be inspected in response.exception()
    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 [anotação `@DynamoDbAttribute\(NAME\)`](#) e forneça o nome desejado como parâmetro.

Executar operações

Depois que a tabela for criada, use a instância `DynamoDbTable` 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 dados `Customer`](#) 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 objeto `customer` para o DynamoDB, registre a saída do método `toString()` do objeto para compará-la com o que é enviado pelo cliente aprimorado.

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

O registro em log em nível de conexão mostra a carga útil da solicitação gerada. O cliente aprimorado gerou a representação de nível baixo da classe de dados. O atributo `regDate`, que é um tipo `Instant` em Java, é representado como uma cadeia de caracteres 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 atributos 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 método `DynamoDbEnhanced.table()`. Isso foi feito no exemplo anterior com a seguinte instrução:

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

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

O exemplo a seguir demonstra isso recuperando imediatamente uma instância `Customer` 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 método `table()` deve corresponder ao nome da tabela existente do DynamoDB.

Saiba mais sobre os fundamentos da API do cliente avançado do DynamoDB

Este tópico discute os atributos básicos da API do Cliente Aprimorado do DynamoDB e a compara com a [API do cliente padrão do DynamoDB](#).

Se você não conhece a API do Cliente Aprimorado do DynamoDB, recomendamos que 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 por estatísticas ou estruturas criadas de modo dinâmico.

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 Documento Aprimorado](#) e requer somente uma chave primária digitada estaticamente e funciona com instâncias `EnhancedDocument` para armazenar os elementos de dados. A API de Documento Aprimorado 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 Cliente Aprimorado do DynamoDB 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 compatível 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 de tipo primitivo Java

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

Valores nulos

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

Para solicitações `updateItem`, 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 `ignoreNulls()` para o método `updateItem()`.

```
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 Cliente Aprimorado do DynamoDB

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 aprimorada oferecem a maioria dos atributos disponíveis no cliente padrão do DynamoDB. Eles estão totalmente documentados na Referência da API do AWS SDK for Java 2.x .

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)
        .addGetItem(key1)
        .addGetItem(key2)
        .addGetItem(key3)
        .build()));

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

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

// TransactWriteItems
enhancedClient.transactWriteItems(r -> r.addConditionCheck(customerTable,
    i -> i.key(orderKey)
    .conditionExpression(conditionExpression))
    .addUpdateItem(customerTable, customer)
    .addDeleteItem(customerTable, key));
```

Comparar o Cliente Aprimorado do DynamoDB com o cliente padrão do DynamoDB

As duas APIs de cliente do DynamoDB: [padrão](#) e [aprimorado](#), permitem que você trabalhe com tabelas do DynamoDB para realizar operações de CRUD (criar, ler, atualizar e excluir) em nível de dados. A diferença entre as APIs do cliente está na execução. Usando o cliente padrão, você trabalha diretamente com atributos de dados de nível baixo. A API do cliente aprimorado usa classes Java conhecidas e mapeia a API de nível baixo nos bastidores.

Embora as duas APIs do cliente forneçam 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 e atualizar tabelas. A API do cliente aprimorado fornece 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 tabela `ProductCatalog` usando o cliente padrão e o cliente aprimorado.

Comparar: criar uma tabela usando o cliente do DynamoDB padrão

```
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))
    );
```

Comparar: criar uma tabela usando o Cliente Aprimorado do DynamoDB

```
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 Cliente Aprimorado do DynamoDB 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 Cliente Aprimorado do DynamoDB. 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('\n');
    sb.append(", isbn=").append(isbn).append('\n');
    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 de uma gravação em lote a seguir ilustram a verbosidade e a falta de segurança de tipo ao usar o cliente padrão em vez do cliente aprimorado.

Comparar: operação de gravação em lote usando o cliente DynamoDB padrão

```
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());
}
```

Comparar: operação de gravação em lote usando o Cliente Aprimorado do DynamoDB

```
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());
}
```

Trabalhar com classes de dados imutáveis

O atributo de mapeamento da API do Cliente Aprimorado do DynamoDB 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 anotação `@DynamoDbBean` conforme

mostrado na [classe Customer](#), as classes imutáveis usam a anotação `@DynamoDbImmutable`, a qual 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 anotar uma classe de dados com `@DynamoDbImmutable`.

1. Todo método que não é uma substituição de `Object.class` e não foi anotado com `@DynamoDbIgnore` deve ser um getter 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 estrutura precisa 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 que retorna uma instância da classe builder. Essa opção é mostrada na classe imutável `Customer`.
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 método `fromImmutableClass()` no `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 `createTable()` do `DynamoDbTable` conforme mostrado no exemplo de trecho 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));
}
```

Usar bibliotecas de terceiros, como Lombok

Bibliotecas de terceiros, como [Project Lombok](#), ajudam a gerar código clichê associado a objetos imutáveis. A API do Cliente Aprimorado do DynamoDB funciona com essas bibliotecas, desde que as classes de dados sigam as convenções detalhadas nesta seção.

O exemplo a seguir mostra a classe `CustomerImmutable` imutável com anotações do Lombok. Observe como o atributo `onMethod` do Lombok copia anotações do DynamoDB baseadas em atributos, como `@DynamoDbPartitionKey`, no código gerado.

```
@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 createdAt;
}
```

Usar expressões e condições

As expressões na API do Cliente Aprimorado do DynamoDB são representações Java das [expressões do DynamoDB](#).

A API do Cliente Aprimorado do DynamoDB usa três tipos de expressões:

[Expressão](#)

A classe `Expression` 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 do seguinte:

- Uma expressão em cadeia de caracteres (obrigatório). Uma cadeia de caracteres 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 objeto `Expression` 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 dos espaços reservados na expressão da cadeia 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 classe [AttributeValues](#) tem métodos convenientes para gerar uma instância `AttributeValue` a partir de um literal. Como alternativa, você pode usar o `AttributeValue.Builder` para gerar uma instância `AttributeValue`.

O trecho a seguir mostra um mapa com duas entradas após a linha de comentário 2. A cadeia de caracteres passada para o método `expression()`, mostrada após a linha de comentário 1, contém os espaços reservados que o DynamoDB determina antes de realizar a operação. Esse trecho não contém um mapa dos nomes das expressões, porque `price` é um nome de atributo permissível.

```
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 a `Expression`.

Por exemplo, se o nome do atributo fosse `1price` em vez de `price` no 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", "price") )
        .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 método `expressionNames()`. 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 da cadeia de caracteres. Um exemplo de função de expressão é `attribute_exists(<attribute_name>)`.

O exemplo a seguir cria uma `Expression` que usa uma [função do DynamoDB](#). Uma cadeia de caracteres de expressão neste exemplo não usa espaços reservados. Essa expressão pode ser usada em uma operação `putItem` para verificar se um item já existe no banco de dados com um valor de atributo `movie` igual ao atributo `movie` do objeto de dados.

```
Expression exp = Expression.builder().expression("attribute_not_exists
(movie)").build();
```

O Guia do desenvolvedor do DynamoDB contém informações completas sobre as expressões [de nível baixo](#) usadas com o DynamoDB.

Expressões de condição e condicionais

Ao usar os métodos `putItem()`, `updateItem()`, `deleteItem()` e também ao usar operações de transação e em lote, você usa objetos de [Expression](#) para especificar as condições que o DynamoDB deve atender para continuar com a operação. Essas expressões são chamadas de expressões de condição. Para ver um exemplo, consulte a expressão de condição usada no método `addDeleteItem()` (após a linha de comentário 1) do [exemplo de transação](#) mostrado neste guia.

Quando você trabalha com os métodos `query()`, uma condição é expressa como uma [QueryConditional](#). A classe `QueryConditional` 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 seção [the section called “Exemplos de métodos Query”](#) 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, de modo que o custo de leitura é o mesmo que se não houvesse filtro. O Guia do desenvolvedor do Amazon DynamoDB 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 verificação. O critério restringe os itens devolvidos a itens com um preço de 8,00 a 80,00.

```
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 `updateItem()` do Cliente Aprimorado do DynamoDB fornece uma forma padrão de atualizar itens no DynamoDB. No entanto, quando você precisa de mais funcionalidades, [UpdateExpressions](#) fornece 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 método `updateItem()`.

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 [Guia do desenvolvedor do Amazon DynamoDB](#).

Trabalhar com resultados paginados: verificações e consultas

Os métodos `scan`, `query` e `batch` da API do Cliente Aprimorado do DynamoDB 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.

Verificar uma tabela

O método do SDK [scan](#) corresponde à [operação do DynamoDB](#) com o mesmo nome. A API do Cliente Aprimorado do DynamoDB 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](#).

Usar a API síncrona

O exemplo a seguir mostra o método `scan` 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 itens `ProductCatalog` que são retornados àqueles com um valor de preço de 8,00 a 80,00.

Este exemplo também exclui os valores `isbn` usando o método `attributesToProject` mostrado após a linha de comentário 2.

Na linha de comentário 3, o objeto `PageIterable`, `pagedResult`, é retornado pelo método `scan`. O método `stream` de `PageIterable` retorna um objeto [java.util.Stream](#), 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 itens `ProductCatalog`. 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 interface `PageIterable` oferece várias maneiras de processar resultados por causa de suas duas interfaces principais: [java.lang.Iterable](#) e [SdkIterable](#). `Iterable` traz os métodos `forEach`, `iterator` e `splititerator`, e `SdkIterable` traz o método `stream`.

```
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())
    );
}
```

Usar a API assíncrona

O método `scan` assíncrono retorna os resultados como um objeto `PagePublisher`. A interface `PagePublisher` tem dois métodos `subscribe` que você pode usar para processar páginas de resposta. Um método `subscribe` vem da interface principal `org.reactivestreams.Publisher`. Para processar páginas usando essa primeira opção, transmita uma instância [Subscriber](#) para o método `subscribe`. O primeiro exemplo a seguir mostra o uso do método `subscribe`.

O segundo `subscribe` método vem da [SdkPublisher](#) interface. Esta versão de `subscribe` aceita um [Consumer](#) em vez de um `Subscriber`. Essa variação do método `subscribe` é mostrada no segundo exemplo.

O exemplo a seguir mostra a versão assíncrona do método `scan` que usa a mesma expressão de filtro mostrada no exemplo anterior.

Após a linha de comentário 3, `DynamoDbAsyncTable.scan` retorna um objeto `PagePublisher`. Na próxima linha, o código cria uma instância da interface `org.reactivestreams.Subscriber`, `ProductCatalogSubscriber`, que assina a `PagePublisher` após o comentário na linha 4.

O objeto `Subscriber` coleta os itens `ProductCatalog` de cada página no método `onNext` após a linha de comentário 8 no exemplo da classe `ProductCatalogSubscriber`. Os itens são armazenados na variável `List` privada e acessados no código de chamada com o método `ProductCatalogSubscriber.getSubscribedItems()`. A chamada é feita após a linha de comentários 5.

Depois que a lista é recuperada, o código classifica todos os itens `ProductCatalog` 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) {
    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 método `PagePublisher.subscribe` que aceita um `Consumer` após a linha de comentário 6. O parâmetro `lambda` em 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 método `items` do `PagePublisher` desempacota 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 método [`query\(\)`](#) da classe `DynamoDbTable` encontra itens com base nos valores das chaves primárias. A anotação `@DynamoDbPartitionKey` e a anotação `@DynamoDbSortKey` opcional são usadas para definir a chave primária em sua classe de dados.

O método `query()` 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 `query()` funcionam do mesmo modo. Assim como na API `scan`, a API `query` 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 verificação.

Exemplos de métodos **Query**

O exemplo de código do método `query()` a seguir usa a classe `MovieActor`. A classe de dados define uma chave primária composta constituída pelo atributo **movie** para a chave de partição e pelo atributo **actor** para a chave de classificação.

A classe também sinaliza que usa um índice secundário global chamado **acting_award_year**. A chave primária composta do índice é constituída pelo atributo **actingaward** para a chave de partição e pelo **actingyear** para chave de classificação. Posteriormente neste tópico, quando mostrarmos como criar e usar índices, nos referiremos ao índice **acting_award_year**.

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.actingAward = actingAward;
    }

    @DynamoDbSecondarySortKey(indexNames = {"acting_award_year", "movie_year"})
    @DynamoDbAttribute("actingyear")
    public Integer getActingYear() {
        return actingYear;
    }

    public void setActingYear(Integer actingYear) {
        this.actingYear = 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 tabela **MovieActor**

```

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 instância `keyEqual` 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 tenha **actingschoolname** ativado 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.
        );
}

```

Esta é a saída gerada pela execução do método. A saída exibe itens com um valor `movieName` de `movie01` e não exibe nenhum item com `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 `keyEqual QueryConditional` pelo `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 instâncias `QueryConditional` exigem um valor de chave de partição. Métodos `QueryConditional` que começam com `sort...` indicam que uma chave de classificação é necessária. Os resultados não são classificados.

A saída a seguir exhibe os resultados da consulta. A consulta retorna itens que têm um valor `movieName` igual a `movie01` e somente itens que têm um valor `actorName` maior ou igual a `actor2`. Como o filtro foi removido, a consulta retorna itens que não têm valor para o atributo `actingSchoolName`.

```
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'}
```

Realizar operações em lote

A API do Cliente Aprimorado do DynamoDB oferece dois métodos em lote, [batchGetItem\(\)](#) e [batchWriteItem\(\)](#).

Exemplo de `batchGetItem()`

Com o método [`DynamoDbTable.batchGetItem\(\)`](#), você pode recuperar até 100 itens individuais em várias tabelas com uma solicitação geral. O exemplo a seguir usa as classes de dados [Customer](#) e [MovieActor](#) mostradas anteriormente.

No exemplo após as linhas 1 e 2, você cria objetos [ReadBatch](#) que são adicionados posteriormente como parâmetros ao método `batchGetItem()` após a linha de comentário 3. O código após a linha de comentário 1 cria o lote para ser lido na tabela `Customer`. O código após a linha de comentário 1a mostra o uso de um construtor [GetItemEnhancedRequest](#) 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 após 2a, você também pode especificar que o DynamoDB deve realizar uma [leitura altamente consistente](#). Quando o método `consistentRead()` é usado, ele deve ser usado em todos os itens solicitados para a mesma tabela.

Para recuperar os itens encontrados pelo DynamoDB, use o método [resultsForTable\(\)](#) mostrado após 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 método `java.util.List`. Este exemplo registra cada item.

Para descobrir itens que o DynamoDB não processou, use a abordagem após a linha de comentários 5. A classe `BatchGetResultPage` tem o método [unprocessedKeysForTable\(\)](#) 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.
```

```

        .addItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
        // 1b. Alternatively, supply a data class instances to provide the
primary key values.
        .addItem(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.
        .addItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")))
        .consistentRead(Boolean.TRUE))
        .addItem(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(movieActorTable).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 método `batchWriteItem()` 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 modelo [ProductCatalog](#) e [MovieActor](#) mostradas anteriormente.

Os objetos `WriteBatch` são construídos após as linhas de comentário 1 e 2. Para a tabela `ProductCatalog`, o código coloca um item e exclui um item. Para a tabela `MovieActor` após a linha de comentário 2, o código coloca dois itens e exclui um.

O método `batchWriteItem` é chamado após a linha de comentário 3. O parâmetro [builder](#) fornece as solicitações em lote para cada tabela.

O objeto [BatchWriteResult](#) 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 colocar e excluir.

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 tabela `MovieActor` que o item do filme `Blue Jasmine` foi substituído pelo item usado na solicitação de colocação adquirida 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 filme `Blue Jasmine`.

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 [colocar](#) e [excluir](#), 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.

Realizar operações de transação

A API do Cliente Aprimorado do DynamoDB fornece o `transactGetItems()` e os métodos `transactWriteItems()`. Os métodos de transação do SDK para Java fornecem atomicidade, consistência, isolamento e durabilidade (ACID) nas tabelas do DynamoDB, ajudando você a manter a correção dos dados em suas aplicações.

Exemplo de `transactGetItems()`

O método [transactGetItems\(\)](#) aceita até 100 solicitações individuais de itens. Todos os itens são lidos em uma única transação atômica. O Guia do desenvolvedor do Amazon DynamoDB tem informações sobre as [condições que causam a falha do método transactGetItems\(\)](#) e também sobre o nível de isolamento usado ao chamar [transactGetItem\(\)](#).

Depois da linha de comentário 1 no exemplo a seguir, o código chama o método `transactGetItems()` com um parâmetro [builder](#). O [addGetItem\(\)](#) do construtor é chamado três vezes com um objeto de dados contendo os valores-chave que o SDK usará para gerar a solicitação final.

A solicitação retorna uma lista de objetos [Document](#) após a linha de comentário 2. A lista de documentos que é retornada contém instâncias de [Documentos](#) não nulas dos dados do item na mesma ordem solicitada. O método [Document.getItem\(MappedTableResource<T> mappedTableResource\)](#) converte um objeto não digitado em um objeto `Document` Java digitado se os dados do item forem retornados, caso contrário, o método retornará nulo.

```
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
    .addGetItem(catalogTable,
Key.builder().partitionValue(2).sortValue("Title 55").build())
    .addGetItem(movieActorTable, Key.builder().partitionValue("Sophie's
Choice").sortValue("Meryl Streep").build())
    .addGetItem(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 retornado, 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 Guia do desenvolvedor do Amazon DynamoDB 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](#) e [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 método `addPutItem()`. O método aceita um objeto [MappedTableResource](#) e a instância do objeto de dados onde colocar. A declaração após a linha de comentário 2 mostra a variação que aceita uma instância [TransactPutItemEnhancedRequest](#). 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 método `ignoreNulls()` que permite configurar o que o SDK faz com os valores `null` no objeto modelo. Se o método `ignoreNulls()` retornar `true`, o SDK não removerá os valores dos atributos da tabela para os atributos do objeto de dados que são `null`. Se o método `ignoreNulls()` 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 afirmação 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, getProductCatId2())
        // 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 parâmetros add*Item.

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 valor `price` é o único valor que foi alterado no item. Se `ignoreNulls()` tivesse retornado `false`, a linha 4 ficaria como 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 executar uma verificação de condição e também tentar atualizar o item na mesma transação.

O exemplo mostra um tipo de cada operação em uma solicitação transacional de itens de gravação. Depois da linha de comentário 2, o método `addConditionCheck()` fornece a condição que falha na transação se o parâmetro `conditionExpression` 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` não é igual a 1982. Se for igual, a expressão será avaliada como `false` e a transação falhará.

Este guia discutirá [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 valor `actingYear` do filme `Sophie's Choice` é 1982, conforme mostrado na linha 2 dos itens na tabela antes da chamada do método `transactWriteItem()`.

Para capturar as informações de cancelamento da transação, inclua a chamada do método `transactWriteItems()` em um bloco `try` e `catch` o [TransactionCanceledException](#). Depois da linha de comentário 4 do exemplo, o código registra cada objeto [CancellationReason](#). 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 filme `Sophie's Choice`.

```
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 valor `actingYear` do filme `Blue Jasmine` é `2013` como mostrado na linha 2 da lista de itens antes da execução do código de exemplo.

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)
```

Usar índices secundários

Os índices secundários melhoram o acesso aos dados definindo chaves alternativas que você usa nas operações de consulta e verificação. Índices secundários globais (GSI) têm uma chave de partição e uma chave de classificação que podem ser diferentes daquelas contidas na tabela base. Por outro lado, os índices secundários locais (LSI) usam a chave de partição do índice primário.

Anotar a classe de dados com anotações de índice secundário

Os atributos que participam de índices secundários exigem a anotação `@DynamoDbSecondaryPartitionKey` ou `@DynamoDbSecondarySortKey`.

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 atributo `Subject` tem uma função dupla. É a chave de classificação da chave primária e a chave de partição do GSI nomeado `SubjectLastPostedDateIndex`.

Classe `MessageThread`

A classe `MessageThread` é adequada para ser usada como uma classe de dados para a [tabela Thread de exemplo](#) no Guia de desenvolvedor do Amazon DynamoDB.

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 +
        '}';
}
}
```

Criar o índice

A partir da versão 2.20.86 do SDK para Java, o método `createTable()` 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 throughput 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 tabela `Thread`. 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 método `indexName()` do construtor 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 terminam nos dois índices após as linhas de comentário 3 e 4. Mais informações sobre [projeções de atributos](#) estão disponíveis no Guia do desenvolvedor do Amazon DynamoDB.

```
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))
        )
    );
}
```

Consultar usando um índice

O exemplo a seguir consulta o índice secundário local `ForumLastPostedDateIndex`.

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 método `query()` no índice que passa o objeto `QueryConditional`.

Você também configura a consulta para retornar três valores de atributos, conforme mostrado após a linha de comentário 5. Se `attributesToProject()` não for chamado, 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, revise os resultados e registre cada item retornado pela consulta e os 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.
    final SdkIterable<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 valor `forumName` de `Forum02` e um valor `lastPostedDateTime` maior ou igual a `2023.03.31`. Os resultados mostram valores `message` com uma cadeia de caracteres vazia, embora os atributos `message` tenham valores no índice. Isso ocorre porque o atributo da mensagem não foi projetado após a linha de comentário 5.

Usar recursos avançados de mapeamento

Saiba mais sobre os atributos avançados do esquema de tabelas na API do Cliente Aprimorado do DynamoDB.

Entender os tipos de esquema de tabela

[TableSchema](#) é a interface para a funcionalidade de mapeamento da API do Cliente Aprimorado do DynamoDB. Ela pode mapear um objeto de dados de e para um mapa de [AttributeValues](#). Um objeto `TableSchema` precisa conhecer a estrutura da tabela que está mapeando. Essas informações de estrutura são armazenadas em um objeto [TableMetadata](#).

A API de cliente aprimorado tem várias implementações de `TableSchema`, conforme será visto a seguir.

Esquema de tabela gerado a partir de classes anotadas

Criar uma `TableSchema` a partir de classes anotadas é uma operação moderadamente dispendiosa, 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 Conceitos básicos](#).

Note

Se uma `BeanTableSchema` não 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. Esse método é descrito na seção [???](#).

Esquema de tabela gerado com um construtor

As seguintes `TableSchemas` são criadas 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 padrões de nomenclatura `JavaBean`.

[StaticTableSchema](#)

Essa implementação foi criada para classes de dados mutáveis. A seção de introdução deste guia demonstrou como [gerar uma `StaticTableSchema` usando um construtor](#).

[StaticImmutableTableSchema](#)

Da mesma forma que você cria um `StaticTableSchema`, você gera uma implementação desse tipo de `TableSchema` usando um [construtor](#) para uso com classes de dados imutáveis.

Esquema de tabela para dados sem um esquema fixo

[DocumentTableSchema](#)

Ao contrário de outras implementações de `TableSchema`, você não define atributos para uma instância `DocumentTableSchema`. Normalmente, você especifica somente chaves primárias e provedores de conversão de atributos. Uma instância `EnhancedDocument` fornece os atributos que você cria a partir de elementos individuais ou de uma cadeia de caracteres `JSON`.

Inclua ou exclua atributos explicitamente

A API do Cliente Aprimorado do `DynamoDB` 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 `@DynamoDbAttribute` e 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 atributo 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 implementação `AttributeConverterProvider` personalizada. Você também pode alterar o conversor para um único atributo.

Para obter uma lista dos conversores disponíveis, consulte o documento Java da interface [AttributeConverter](#).

Forneça provedores de conversão de atributos personalizados

Você pode fornecer um único `AttributeConverterProvider` ou uma cadeia de `AttributeConverterProviders` ordenados por meio da anotação `@DynamoDbBean` (`converterProviders = {...}`). Qualquer personalização do `AttributeConverterProvider` deve estender a interface do `AttributeConverterProvider`.

Observe que, se fornecer sua própria cadeia de provedores de conversão de atributos, você substituirá o provedor de conversão padrão, `DefaultAttributeConverterProvider`. Se quiser usar a funcionalidade do `DefaultAttributeConverterProvider`, você deverá 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 menor prioridade.

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {

}
```

Os criadores de esquemas de tabelas estáticas têm um método `attributeConverterProviders()` 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 `AttributeConverterProviders` no esquema da tabela. A ação 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 anotação `@DynamoDbConvertedBy` é usada para especificar a classe `AttributeConverter` 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 de atributos `attributeConverter()` equivalente. Esse método usa uma instância de um `AttributeConverter`, 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 implementação `AttributeConverterProvider` que fornece um conversor de atributos para objetos [java.net.HttpCookie](http://java.net/HttpCookie).

A classe `SimpleUser` a seguir contém um atributo chamado `lastUsedCookie` que é uma instância de `HttpCookie`.

O parâmetro para as anotações `@DynamoDbBean` lista as duas classes `AttributeConverterProvider` 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` no exemplo a seguir fornece uma instância de um `HttpCookieConverter`.

```
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 método `transformFrom()` da classe `HttpCookieConverter` a seguir, o código recebe uma instância `HttpCookie` e a transforma em um mapa do DynamoDB que é armazenado como um atributo.

O método `transformTo()` recebe um parâmetro de mapa do DynamoDB e, em seguida, invoca o construtor `HttpCookie` 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 do Cliente Aprimorado 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 o valor seja gravado somente se ainda não houver nenhum valor existente para o atributo no banco de dados. Nesse caso, você usa o comportamento de atualização [WRITE_IF_NOT_EXISTS](#).

O exemplo a seguir mostra a anotação que adiciona o comportamento ao atributo `createdOn`.

```

@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 Cliente Aprimorado do DynamoDB fornece suporte para nivelar os atributos em uma classe.

Usar herança

Se suas classes usam herança, use as seguintes abordagens para nivelar a hierarquia.

Usar beans anotados

Para a abordagem de anotação, ambas as classes devem conter a anotação `@DynamoDbBean` 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 createdAt;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

Lombok

A [opção onMethod](#) do Lombok copia anotações do DynamoDB baseadas em atributos, como `@DynamoDbPartitionKey`, no código gerado.

```

@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 createdAt;
}

```

Usar esquemas estáticos

Para a abordagem do esquema estático, use o método `extend()` do construtor para reduzir os atributos da classe principal para a classe secundária. Isso é mostrado depois da linha de comentário 1 no seguinte exemplo:

```
    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: Uma vez que 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 createdAt;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
```

Lombok

```
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdAt;
}
```

Usar composição

Se suas classes usam composição, use as seguintes abordagens para nivelar a hierarquia.

Usar beans anotados

A anotação `@DynamoDbFlatten` nivela a classe contida.

Os exemplos de classes de dados a seguir usam a anotação `@DynamoDbFlatten` para adicionar efetivamente todos os atributos da classe `GenericRecord` contida à classe `Customer`.

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 createdAt;

        @DynamoDbPartitionKey
        public String getId() { return this.id; }
        public void setId(String id) { this.id = id; }

        public String getCreatedAt() { return this.createdAt; }
        public void setCreatedAt(String createdAt) { this.createdAt =
        createdAt; }
    }
}
```

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 createdAt;
}
```

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.

Usar esquemas estáticos

Ao criar um esquema de tabela estático, use o método `flatten()` 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::getCreatedAt)
            .setter(GenericRecord::setCreatedAt))
        .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 createdAt;

    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return this.createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

Lombok

```

@Data
public class Customer {
    private String name;
    private GenericRecord record;

```

```

}

@Data
public class GenericRecord {
    private String id;
    private String createdAt;
}

```

Você pode usar o padrão do construtor para nivelar quantas classes elegíveis forem necessárias.

Implicações para outro código

Quando você usa o atributo `@DynamoDbFlatten` (ou método `flatten()` do construtor), o item no DynamoDB contém um atributo para cada atributo do objeto composto. Isso 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 classe `Customer` mostrada no [exemplo de nivelamento com composição](#) com e sem nivelamento do atributo `record`. Você pode visualizar a diferença com JSON conforme mostrado na tabela a seguir.

Com nivelamento	Sem nivelamento
3 atributos	2 atributos
<pre> { "id": "1", "createdAt": "today", "name": "my name" } </pre>	<pre> { "id": "1", "record": { "createdAt": "today", "name": "my name" } } </pre>

A diferença se torna importante se você tiver outro código acessando a tabela do DynamoDB que espera encontrar certos atributos.

Trabalhar 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 uma `List` ou `Map`. Também corresponde a uma instância de um tipo complexo, como `Address` ou `PhoneNumber`, conforme usado na classe `Person` 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

Usar classes anotadas

Você pode salvar atributos aninhados para classes personalizadas fazendo anotações neles. A classe `Address` e a classe `PhoneNumber` mostradas anteriormente são anotadas somente com a anotação `@DynamoDbBean`. Quando a API do Cliente Aprimorado do DynamoDB cria o esquema de tabela para a classe `Person` com o seguinte trecho, a API descobre o uso das classes `Address` e `PhoneNumber` e cria os mapeamentos correspondentes para funcionar com o DynamoDB.

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

Usar 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 classes `Address` e `PhoneNumber` são abstratos, uma vez 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` no método `EnhanceType.documentOf(...)` não indica que o método retorne um tipo `EnhancedDocument` da API de documento aprimorado. O método `documentOf(...)` 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(
            // 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();

```

Projetar atributos aninhados

Para os métodos `query()` e `scan()`, você pode especificar quais atributos você deseja que sejam retornados nos resultados usando chamadas de método como `addNestedAttributeToProject()` e `attributesToProject()`. A API do Cliente Aprimorado do DynamoDB converte os parâmetros de chamada do método Java em [expressões de projeção](#) antes do envio da solicitação.

O exemplo a seguir preenche a tabela `Person` com dois itens e, em seguida, executa três operações de verificação.

A primeira verificação acessa todos os itens da tabela para comparar os resultados com as outras operações de verificação.

A segunda verificação usa o método do construtor [addNestedAttributeToProject\(\)](#) para retornar somente o valor do atributo `street`.

A terceira operação de varredura usa o método do construtor [attributesToProject\(\)](#) para retornar os dados do atributo de primeiro nível, `hobbies`. O tipo do atributo de `hobbies` é uma lista. Para acessar itens individuais da lista, execute uma operação `get()` 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 método `attributesToProject()` 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 verificação realizada com a instância `ScanEnhancedRequest` no trecho a seguir retorna somente dados de 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 método `attributesToProject()` primeiro. Essa ordenação preserva todos os outros atributos obrigatórios.

```
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]}

```

Preservar 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.

Exemplos de beans

A classe de dados a seguir contém dois campos `InnerBean`. O método `getter`, `getInnerBeanWithoutAnno()`, não é anotado com `@DynamoDbPreserveEmptyObject`. O método `getInnerBeanWithAnno()` 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 classe `InnerBean` a seguir são campos de `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 objeto MyBean 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 preserveEmptyObjectAnnoUsingGetItemExample(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 versão `StaticTableSchema` 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();
}

```

Evitar 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.

Exemplos de beans

A classe de dados a seguir contém dois campos `InnerBean`. O método `getter`, `getInnerBeanWithoutAnno()`, não é anotado. O método `getInnerBeanWithIgnoreNullsAnno()` 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 classe `InnerBean` 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 objeto `InnerBean` 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 nível baixo 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 instância `innerBeanWithoutAnno` 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 atributo

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 versão `StaticTableSchema` 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();
}
```

Trabalhar com documentos JSON com a API de documentos aprimorados do DynamoDB

A [API de Documento Aprimorado](#) para AWS SDK for Java 2.x foi projetada para funcionar com dados orientados a documentos que não têm um esquema fixo. No entanto, ela também permite que você use classes personalizadas para mapear atributos individuais.

A API de Documento Aprimorado é a sucessora da [API de Documentos](#) da AWS SDK for Java v1.x.

Sumário

- [Começar a usar a API de documentos aprimorados](#)
 - [Criar um DocumentTableSchema e uma DynamoDbTable](#)
- [Criar documentos aprimorados](#)
 - [Criar a partir de uma cadeia de caracteres JSON](#)
 - [Criar a partir de elementos individuais](#)
- [Executar operações de CRUD](#)
- [Acessar atributos do documento aprimorado como objetos personalizados](#)
- [Usar um EnhancedDocument sem o DynamoDB](#)

Começar a usar a API de documentos aprimorados

A API de Documento Aprimorado exige as mesmas [dependências](#) necessárias para a API do Cliente Aprimorado do DynamoDB. Ela também requer uma [instância DynamoDbEnhancedClient](#), conforme mostrado no início deste tópico.

Como a API de Documento Aprimorado foi lançada com a versão 2.20.3 do AWS SDK for Java 2.x, você precisa dessa versão ou uma superior.

Criar um `DocumentTableSchema` e uma `DynamoDbTable`

Para invocar comandos em uma tabela do DynamoDB usando a API de Documento Aprimorado, associe a tabela a um objeto de recurso do [DynamoDbTable <EnhancedDocument>](#) do lado do cliente.

O método `table()` do cliente aprimorado cria uma instância `DynamoDbTable<EnhancedDocument>` 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 método `AttributeConverterProvider.defaultProvider()` 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 `person` do DynamoDB que armazena objetos `EnhancedDocument` sem esquema.

```
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.
```

Veja a seguir a representação JSON de um objeto `person` usado em toda esta 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"
  }
]
}
```

Criar documentos aprimorados

Um [EnhancedDocument](#) representa um objeto do tipo documento que tem uma estrutura complexa com atributos aninhados. Um 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 instância `EnhancedDocument` usando um construtor que fornece várias maneiras de adicionar elementos.

Criar a partir de uma cadeia de caracteres JSON

Com uma cadeia de caracteres JSON, você pode criar um `EnhancedDocument` com uma chamada do método. O trecho a seguir cria um `EnhancedDocument` a partir de uma cadeia de caracteres JSON retornada pelo método auxiliar `jsonPerson()`. O método `jsonPerson()` retorna a versão da cadeia de caracteres JSON do [objeto person](#) mostrado anteriormente.

```
EnhancedDocument document =
    EnhancedDocument.builder()
        .json( jsonPerson() )
        .build();
```

Criar a partir de elementos individuais

Como alternativa, você pode criar uma instância `EnhancedDocument` a partir de componentes individuais usando métodos seguros de tipo do construtor.

O exemplo a seguir cria um documento aprimorado `person` semelhante ao documento aprimorado criado a partir da cadeia de caracteres 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"));
}

```

```
}
```

Executar operações de CRUD

Depois que definir uma instância `EnhancedDocument`, você poderá 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 aprimorado a partir 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 cadeia de caracteres 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"}]}

```

Instâncias EnhancedDocument podem ser usadas com qualquer método do [DynamoDbTable](#) ou com o [DynamoDbEnhancedClient](#) no lugar das classes de dados mapeadas.

Acessar atributos do documento aprimorado como objetos personalizados

Além de fornecer uma API para ler e gravar atributos com estruturas sem esquemas, a API de Documento Aprimorado permite converter atributos de e para instâncias de classes personalizadas.

A API de Documento Aprimorado usa AttributeConverterProviders e AttributeConverters que foram mostrados na seção de [conversão de atributos de controle](#) como parte da API do Cliente Aprimorado do DynamoDB.

No exemplo a seguir, usamos um CustomAttributeConverterProvider com sua classe AddressConverter aninhada para converter objetos Address.

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 objetos Address 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 getAddressForCustomMappingExample() 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 = getAddressForCustomMappingExample();

    // 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'}

```

Código CustomAttributeConverterProvider

```

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 instâncias `Address` personalizadas para valores em vez de instâncias `Map<String, String>` 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);
}
```

Usar um **EnhancedDocument** sem o DynamoDB

Embora você geralmente use uma instância de um `EnhancedDocument` para ler e gravar itens do tipo documento do DynamoDB, ela também pode ser usada independentemente do DynamoDB.

Você pode usar `EnhancedDocuments` pela sua capacidade de converter desde cadeias de caracteres JSON ou objetos personalizados a mapas de nível baixo de `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)}
    addressAsJsonString: {"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, defina 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.

Usar extensões

A API do Cliente Aprimorado do DynamoDB fornece suporte a extensões de complemento que fornecem funcionalidades além das operações de mapeamento. As extensões têm dois métodos de hook, `beforeWrite()` e `afterRead()`. `beforeWrite()` modifica uma operação de gravação antes que ela aconteça, e o método `afterRead()` modifica os resultados de uma operação de leitura depois que ela acontece. Uma vez que algumas operações (como atualizações de itens) realizam uma gravação e depois uma leitura, os dois métodos de hook são chamados.

As extensões são carregadas na ordem em que são especificadas no construtor do cliente 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 aprimorado vem com um conjunto de extensões de complemento localizadas no pacote de [extensions](#). Por padrão, o cliente aprimorado carrega a [VersionedRecordExtension](#) e a [AtomicCounterExtension](#). Você pode substituir o comportamento padrão com o construtor de cliente avançado 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 uma 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. A `AtomicCounterExtension` não está carregada neste exemplo.

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
            verifyChecksumExtension)
        .build();
```

VersionedRecordExtension

A `VersionedRecordExtension` é carregada 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 real persistido não corresponder ao valor que o aplicativo leu pela última vez. Esse comportamento fornece efetivamente um bloqueio positivo para atualizações de itens. Se outro processo atualizar um item entre o momento em que o primeiro processo leu 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 atributo `version` 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

A `AtomicCounterExtension` é carregada 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 de 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

A `AutoGeneratedTimestampRecordExtension` atualiza 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 carimbo de data/hora atual, marque o atributo `Instant` no esquema da tabela.

O atributo `lastUpdate` é o alvo do comportamento das extensões no trecho a seguir. Observe a exigência de que o atributo seja um tipo `Instant`.

```
@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 método `beforeWrite()` que usa uma expressão de atualização. Após a linha de comentário 2, criamos um `SetAction` para definir o atributo `registrationDate` se o item no banco de dados ainda não tiver um atributo `registrationDate`. Sempre que um objeto `Customer` é atualizado, a extensão garante que um `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();
    }
}
```

```
}  
}
```

Usar a API do cliente avançado do DynamoDB de forma assíncrona

Se seu aplicativo exigir chamadas assíncronas e sem bloqueio para o DynamoDB, você poderá usar o [DynamoDbEnhancedAsyncClient](#). Ele é semelhante à implementação síncrona, mas com as seguintes diferenças principais:

1. Ao criar o `DynamoDbEnhancedAsyncClient`, é necessário 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. Entretanto, seu aplicativo pode 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 `DynamoDbEnhancedClient` síncrono retorna para os mesmos métodos. Seu aplicativo pode então inscrever um manipulador nesse publicador 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 a `SdkPublisher` API, consulte [o exemplo](#) na seção que discute o método `scan()` 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. A tabela é classificada em ordem alfabética crescente pelo nome da anotação.

Anotações de classe de dados usadas neste guia

Nome da anotação	A anotação se aplica a*	O que ela faz	Onde ela é mostrada neste guia
DynamoDbAtomicCounter	atributo	Incrementa um atributo numérico marcado sempre que um registro é gravado no banco de dados.	Introdução e discussão.
DynamoDbAttribute	atributo	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 Conceitos básicos: consultar Observação. • Exemplos do método In Query em MovieActor classe.
DynamoDbAutoGeneratedTimestampAttribute	atributo	Atualiza um atributo marcado com um carimbo de data/hora atual toda vez que o item é gravado com sucesso no banco de dados	Introdução e discussão.
DynamoDbBean	classe	Marca uma classe de dados como mapeável para um esquema de tabela.	Primeiro uso na classe Customer na seção Conceitos

Nome da anotação	A anotação se aplica a*	O que ela faz	Onde ela é mostrada neste guia
			básicos. Vários usos aparecem no guia.
<code>DynamoDbConvertedBy</code>	atributo	Associa um <code>AttributeConverter</code> personalizado ao atributo anotado.	Discussão inicial e exemplo.
<code>DynamoDbFlatten</code>	atributo	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 outros códigos.
<code>DynamoDbIgnore</code>	atributo	Faz com que o atributo permaneça não mapeado.	<ul style="list-style-type: none"> • Discussão inicial. • Use na ProductCatalog aula.
<code>DynamoDbIgnoreNulls</code>	atributo	Impede salvar atributos nulos de objetos aninhados DynamoDb .	Discussão e exemplos.
<code>DynamoDbImmutable</code>	classe	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. • Uso com Lombok.

Nome da anotação	A anotação se aplica a*	O que ela faz	Onde ela é mostrada neste guia
DynamoDbPartitionKey	atributo	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 Conceitos básicos. • Com Lombok.
DynamoDbP reserveEmptyObject	atributo	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.
DynamoDbS econdaryPartitionKey	atributo	Marca um atributo como uma chave de partição para um índice secundário global.	<ul style="list-style-type: none"> • Uso em índices secundários e exemplo. • Exemplos do método In Query. • Exemplo do In Lombok • Com classes imutáveis.

Nome da anotação	A anotação se aplica a*	O que ela faz	Onde ela é mostrada neste guia
DynamoDbSecondarySortKey	atributo	Marca um atributo como uma chave de classificação opcional para um índice secundário global ou local.	<ul style="list-style-type: none"> • Uso em índices secundários e exemplo. • Exemplos do método In Query. • Exemplo do In Lombok. • Com classes imutáveis.
DynamoDbSortKey	atributo	Marca um atributo como a chave de classificação primária opcional (chave de intervalo).	<ul style="list-style-type: none"> • Seção Conceitos básicos da classe Customer. • Com classes imutáveis. • Exemplo do In Lombok. • Exemplos do método In Query.
DynamoDbUpdateBehavior	atributo	Especifica o comportamento quando esse atributo é atualizado como parte de uma operação de 'atualização', como UpdateItem.	Introdução e exemplo.
DynamoDbVersionAttribute	atributo	Incrementa o número da versão de um item.	Introdução e discussão.

*Você pode aplicar anotações em nível de atributo ao getter ou setter, mas não a ambos. Este guia mostra anotações sobre getters.

Trabalhe com Amazon EC2

Esta seção fornece exemplos de programação [Amazon EC2](#) que usam o AWS SDK for Java 2.x.

Tópicos

- [Gerenciar instâncias Amazon EC2](#)
- [Zonas Regiões da AWS de uso e disponibilidade](#)
- [Trabalhe com grupos de segurança em Amazon EC2](#)
- [Trabalhar com metadados da instância do Amazon EC2](#)

Gerenciar instâncias Amazon EC2

Criar uma instância

Crie uma nova instância do Amazon EC2 chamando o método [runInstances](#) do [Ec2Client](#), fornecendo um [RunInstancesRequest](#) contendo a [imagem de máquina da Amazon \(AMI\)](#) a ser usada 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](#) no GitHub.

Iniciar uma instância

Para iniciar uma instância do Amazon EC2, chame o método [startInstances](#) do `Ec2Client`, fornecendo um [StartInstancesRequest](#) contendo o ID da instância para iniciar.

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](#) no GitHub.

Interromper uma instância

Para interromper uma instância do Amazon EC2, chame o método [stopInstances](#) do `Ec2Client`, fornecendo um [StopInstancesRequest](#) contendo o ID da instância para interromper.

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](#) no GitHub.

Reinicializar uma instância

Para reinicializar uma instância do Amazon EC2, chame o método [rebootInstances](#) do `Ec2Client`, fornecendo um [RebootInstancesRequest](#) contendo o ID da instância para reinicializar.

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](#) no GitHub.

Descrever instâncias

Para listar as instâncias, crie um [DescribeInstancesRequest](#) e chame o método [describeInstances](#) do `Ec2Client`. Isso retornará um objeto [DescribeInstancesResponse](#), que poderá ser usado para listar as instâncias do Amazon EC2 para a conta e a 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 instâncias 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](#) no GitHub.

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 do Amazon EC2](#) no Guia do Usuário do Amazon EC2 para instâncias do Linux.

Para iniciar o monitoramento de uma instância, você deve criar um [MonitorInstancesRequest](#) com o ID da instância para monitorar e passá-lo para o método [monitorInstances](#) do `Ec2Client`.

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](#) no GitHub.

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 passá-lo para o método [unmonitorInstances](#) do `Ec2Client`.

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](#) no GitHub.

Mais informações

- [RunInstances](#) na Referência de API do Amazon EC2
- [DescribeInstances](#) na Referência de API do Amazon EC2
- [StartInstances](#) na Referência de API do Amazon EC2
- [StopInstances](#) na Referência de API do Amazon EC2
- [RebootInstances](#) na Referência de API do Amazon EC2
- [MonitorInstances](#) na referência de API do Amazon EC2

- [UnmonitorInstances](#) na referência de API do Amazon EC2

Zonas Regiões da AWS de uso e disponibilidade

Descrever regiões

Para listar as regiões disponíveis para a conta, chame o método `describeRegions` do `Ec2Client`. 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.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Código

```
try {
    DescribeRegionsResponse regionsResponse = ec2.describeRegions();
    regionsResponse.regions().forEach(region -> {
        System.out.printf(
            "Found Region %s with endpoint %s%n",
            region.regionName(),
            region.endpoint());
        System.out.println();
    });
}
```

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

Descrever zonas de disponibilidade

Para listar cada zona de disponibilidade disponível para a conta, chame o método `describeAvailabilityZones` do `Ec2Client`. 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.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

Código

Criar o `Ec2Client`.

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

Em seguida, chame `describeAvailabilityZones ()` e recupere os resultados.

```
DescribeAvailabilityZonesResponse zonesResponse =
ec2.describeAvailabilityZones();
zonesResponse.availabilityZones().forEach(zone -> {
    System.out.printf(
        "Found Availability Zone %s with status %s in region %s%n",
        zone.zoneName(),
        zone.state(),
        zone.regionName()
    );
    System.out.println();
});
```

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

Descrever contas

Para listar informações relacionadas ao EC2 sobre sua conta, chame o método `describeAccountAttributes` do `Ec2Client`. 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 `AccountAttribute` objeto.

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.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

```
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;

/**
 * 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 DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Account(ec2);
        System.out.print("Done");
        ec2.close();
    }

    public static void describeEC2Account(Ec2Client ec2) {
        try {
            DescribeAccountAttributesResponse accountResults =
                ec2.describeAccountAttributes();
        }
    }
}
```

```
        accountResults.accountAttributes().forEach(attribute -> {
            System.out.println("\n The name of the attribute is " +
attribute.attributeName());
            attribute.attributeValues().forEach(
                myValue -> System.out.println("\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

Trabalhe com grupos de segurança em Amazon EC2

Criar um grupo de segurança

Para criar um grupo de segurança, chame o `createSecurityGroup` método do `Ec2Client` com um [CreateSecurityGroupRequest](#) que contenha 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](#) em GitHub.

Configurar um grupo de segurança

Um grupo de segurança pode controlar o tráfego de entrada (entrada) e saída (saída) para suas instâncias. Amazon EC2

Para adicionar regras de entrada ao seu grupo de segurança, use o `authorizeSecurityGroupIngress` método do `Ec2Client`, fornecendo o nome do grupo de segurança e as regras de acesso ([IpPermission](#)) que você deseja atribuir a ele dentro de um objeto. [AuthorizeSecurityGroupIngressRequest](#) 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 método `authorizeSecurityGroupIngress` do `Ec2Client`,

```
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 no método do `Ec2Client`. [AuthorizeSecurityGroupEgressRequest](#)`authorizeSecurityGroupEgress`

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

Descrever grupos de segurança

Para descrever os grupos de segurança ou obter informações sobre eles, chame o método `describeSecurityGroups` do `Ec2Client`. Ele retorna um [DescribeSecurityGroupsResponse](#) que você pode usar para acessar a lista de grupos de segurança chamando seu `securityGroups` método, que retorna uma lista de [SecurityGroup](#) objetos.

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](#) em GitHub.

Exclua um grupo de segurança

Para excluir um grupo de segurança, chame o `deleteSecurityGroup` método do `Ec2Client`, passando a ele um [DeleteSecurityGroupRequest](#) que contém o ID do grupo de segurança 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](#) em GitHub.

Mais informações

- [Amazon EC2 Grupos de segurança](#) no guia Amazon EC2 do usuário para instâncias Linux
- [Autorize o tráfego de entrada para suas instâncias Linux](#) no Guia do Amazon EC2 usuário para instâncias Linux
- [CreateSecurityGroup](#) na Referência da Amazon EC2 API
- [DescribeSecurityGroups](#) na Referência da Amazon EC2 API
- [DeleteSecurityGroup](#) na Referência da Amazon EC2 API
- [AuthorizeSecurityGroupIngress](#) na Referência da Amazon EC2 API

Trabalhar 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 à sessão.

Duas classes de cliente estão disponíveis no SDK. O [Ec2MetadataClient](#) síncrono serve 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 um [SdkHttpClient](#) (ou um [SdkAsyncHttpClient](#) para a variante assíncrona) no caminho de classe.

O XML do Maven a seguir mostra trechos de dependência para usar o [URLConnectionHttpClient](#) síncrono junto com a dependência de 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 artefato bom.

Para usar um cliente HTTP assíncrono, substitua o trecho de dependência do artefato `url-connection-client`. Por exemplo, o trecho a seguir traz a implementação do [NettynioAsynchttpClient](#).

```

<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>netty-nio-client</artifactId>
</dependency>

```

Usar o cliente de metadados

Instanciar um cliente de metadados

Você pode instanciar uma instância de um `Ec2MetadataClient` síncrono quando somente uma implementação da interface `SdkHttpClient` está presente no caminho de classe. Para fazer isso, chame o método estático `Ec2MetadataClient#create()`, conforme mostrado no trecho a seguir.

```

Ec2MetadataClient client = Ec2MetadataClient.create(); //
'Ec2MetadataAsyncClient#create' is the asynchronous version.

```

Se seu aplicativo tiver várias implementações da interface `SdkHttpClient` ou `SdkHttpAsyncClient`, você deverá especificar uma implementação para o cliente de metadados usar, conforme mostrado na seção [the section called “Cliente HTTP configurável”](#).

Note

Para a maioria dos clientes de serviços, como o Amazon S3, o SDK for Java adiciona automaticamente implementações da interface `SdkHttpClient` ou `SdkHttpAsyncClient`. Se seu cliente de metadados usar a mesma implementação, o `Ec2MetadataClient#create()` funcionará. Se você precisar de uma implementação diferente, deverá especificá-la ao criar o cliente de metadados.

Envie solicitações

Para recuperar os metadados da instância, instancie a classe `EC2MetadataClient` e chame o método `get` com um parâmetro de caminho que especifica a [categoria de metadados da instância](#).

O exemplo a seguir imprime o valor associado à chave `ami-id` no console.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/ami-id");
System.out.println(response.asString());
client.close(); // Closes the internal resources used by the Ec2MetadataClient class.
```

Se o caminho não for válido, o método `get` gerará uma exceção.

Reutilize a mesma instância do cliente para várias solicitações, mas chame `close` no cliente quando não for mais necessário liberar recursos. Depois que o método de fechamento é chamado, a instância do cliente não pode mais ser usada.

Analisar respostas

Os metadados da instância EC2 podem ser gerados em formatos diferentes. Texto sem formatação e JSON são os formatos mais usados. Os clientes de metadados oferecem formas de trabalhar com esses formatos.

Como mostra o exemplo a seguir, use o método `asString` para obter os dados como uma string Java. Você também pode usar o método `asList` para separar uma resposta de texto sem formatação que retorna 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 método `Ec2MetadataResponse#asDocument` 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 sucesso, 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 em 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 recuo exponencial entre as tentativas.

Se seu caso de uso exigir um mecanismo de repetição diferente, você poderá personalizar o cliente usando o método `retryPolicy` 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();
```

Existem várias [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 do `Ec2MetadataRetryPolicy#none()` desativa a política de repetição padrão para que o cliente de metadados não tente fazer novas tentativas.

Versão de 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 `endpoint` 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 classe `Ec2MetadataAsyncClient`. O código no exemplo a seguir cria um cliente assíncrono com configurações padrão e usa o método `get` para recuperar o valor da chave `ami-id`.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/ami-id");
```

O `java.util.concurrent.CompletableFuture` retornado pelo método `get` é concluído quando a resposta retorna. O exemplo a seguir imprime os metadados `ami-id` no console.

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

Cliente HTTP configurável

O construtor de cada cliente de metadados tem um método `httpClient` que você pode usar para fornecer um cliente HTTP personalizado.

O exemplo a seguir mostra o código para uma instância personalizada do `URLConnectionHttpClient`.

```
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 instância personalizada do `NettyNioAsyncHttpClient` 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 tópico [the section called “Clientes HTTP”](#) deste guia fornece detalhes sobre como configurar os clientes HTTP que estão disponíveis no SDK para Java.

Armazenamento em 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 valor de vida útil padrão, a menos que seu caso de uso específico exija configuração avançada.

Se necessário, configure a duração usando o método `tokenTtl` do construtor. Por exemplo, o código no trecho a seguir cria um cliente com duração de sessão de cinco minutos.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .tokenTtl(Duration.ofMinutes(5))
        .build();
```

Se você omitir a chamada do método `tokenTtl` no construtor, a duração padrão de 21.600 será usada em vez disso.

Trabalhar com IAM

Esta seção apresenta exemplos de como programar o AWS Identity and Access Management (IAM) usando o AWS SDK for Java 2.x.

O AWS Identity and Access Management (IAM) permite que você controle com segurança o acesso aos serviços e recursos da AWS para seus usuários. Usando o IAM, você pode criar e gerenciar usuários e grupos da AWS, além de usar permissões para permitir e negar acesso a recursos da AWS. Para obter um guia completo do IAM, visite o [Guia do usuário do IAM](#).

Os exemplos a seguir incluem somente o código necessário para demonstrar cada técnica. O [código de exemplo completo está disponível no 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 usuários do IAM](#)

- [Crie políticas do IAM com o AWS SDK for Java 2.x](#)
- [Trabalhar com políticas do IAM](#)
- [Trabalhe com certificados de IAM servidor](#)

Gerenciar chaves de IAM acesso

Criar uma chave de acesso

Para criar uma chave de IAM acesso, chame o `IamClient`'s `createAccessKey` método com um [CreateAccessKeyRequest](#) objeto.

Note

Você deve definir a região como `AWS_GLOBAL` para que as `IamClient` chamadas funcionem porque 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 do usuário para o qual listar as chaves e passe-o para o `IamClient`'s `listAccessKeys` método.

Note

Se você não fornecer um nome de usuário `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 `IamClient`'s `getAccessKeyLastUsed` método 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 a última hora usada pela 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 nome desejado e, em seguida [status](#), passando o objeto de solicitação para o `IamClient`'s `updateAccessKey` método.

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 `IamClient`'s `deleteKey` método, fornecendo a ele o ID [DeleteAccessKeyRequest](#) 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](#) em GitHub.

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 usuários do IAM

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](#) em GitHub.

Listar usuários

Para listar os IAM usuários da sua conta, crie uma nova [ListUsersRequest](#) passe-a 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](#) em GitHub.

Excluir um usuário

Para excluir um usuário, chame a `IamClient` `deleteUser` solicitação com um [UpdateUserRequest](#) objeto definido com o nome do 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](#) em GitHub.

Mais informações

- [Usuários IAM](#) no Guia do Usuário IAM
- [Gerenciando usuários IAM](#) no Guia do usuário IAM
- [CreateUser](#) na Referência da IAM API
- [ListUsers](#) na Referência da IAM API
- [UpdateUser](#) na Referência da IAM API
- [DeleteUser](#) na Referência da IAM API

Crie políticas do IAM com o AWS SDK for Java 2.x

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 cadeia de caracteres JSON ou lendo um arquivo, a API fornece uma abordagem orientada a objetos do lado do cliente para gerar a cadeia de caracteres JSON. Quando você lê uma política do IAM existente no formato JSON, a API a converte em uma [IamPolicy](#) instância para manipulação.

A API de criação de política do IAM 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 bloco de dependência de exemplo para um arquivo Maven `pom.xml`. Isso permite que você use a API de criação de política do IAM em seu projeto.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
```

```
<version>2.20.139</version>
</dependency>
```

Criar uma **IamPolicy**

Esta seção mostra vários exemplos de como criar políticas usando a API de criação de política do IAM.

Em cada um dos exemplos a seguir, comece com o [IamPolicy.Builder](#) e adicione uma ou mais instruções usando o método `addStatement`. 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 com base em um período

O exemplo a seguir cria uma política baseada em identidade que permite a ação `GetItem` do Amazon DynamoDB entre dois instantes.

```
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 instrução no exemplo anterior retorna a seguinte cadeia de caracteres JSON.

Leia mais sobre esse [exemplo](#) no Guia do usuário do AWS Identity and Access Management .

```
{
  "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: especificar várias condições

Este exemplo 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 instrução no exemplo anterior retorna a seguinte cadeia de caracteres JSON.

Leia mais sobre esse [exemplo](#) no Guia do usuário do AWS Identity and Access Management .

```

{
  "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: especificar entidades principais

O exemplo a seguir mostra como criar uma política baseada em recursos que negue acesso a um bucket para todas as entidades principais, exceto àquelas 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 instrução no exemplo anterior retorna a seguinte cadeia de caracteres JSON.

Leia mais sobre esse [exemplo](#) no Guia do usuário do AWS Identity and Access Management .

```

{
  "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 o 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 instrução no exemplo anterior retorna a seguinte cadeia de caracteres 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"
      }
    }
  }
}

```

Usar um **IamPolicy** com o IAM

Depois de criar uma instância `IamPolicy`, você usará 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 cadeia de caracteres 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 instrução. 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 argumento `IamClient` 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"
  }
}

```

Trabalhar com políticas do IAM

Criar uma política

Para criar uma nova política, forneça o nome da política e um documento de política formatado em JSON em um 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 [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 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](#) em GitHub.

Listar políticas de função anexadas

Liste as políticas anexadas em uma função chamando `IamClient` o `listAttachedRolePolicies` método s. É necessário 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](#) em GitHub.

Desanexar uma política de função

Para separar uma política de uma função, chame o `detachRolePolicy` método `IamClient`'s, fornecendo 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](#) em GitHub.

Mais informações

- [Visão geral das políticas IAM](#) no Guia do usuário IAM.
- [Referência da política do IAM da AWS](#) no Guia do usuário do IAM.
- [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 de IAM servidor

Para habilitar conexões HTTPS com seu site ou aplicativo AWS, você precisa de um certificado de servidor SSL/TLS. Você pode usar um certificado de servidor fornecido por AWS Certificate Manager ou obtido de um provedor externo.

Recomendamos que você use ACM para provisionar, gerenciar e implantar seus certificados de servidor. Com ACM você pode solicitar um certificado, implantá-lo em seus AWS recursos e deixar que você ACM gerencie as renovações de certificados. Os certificados fornecidos pela ACM são gratuitos. Para obter mais informações sobre ACM, consulte o [Guia AWS Certificate Manager do usuário](#).

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.getServerCertificate().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 [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 `IamClient` o `updateServerCertificate` método s. É necessário um [UpdateServerCertificateRequest](#) objeto definido com o nome atual do certificado do servidor e um novo nome ou novo caminho para ser usado.

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](#) em GitHub.

Excluir um certificado do servidor

Para excluir um certificado de servidor, chame o `deleteServerCertificate` método `IamClient`'s [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](#) em GitHub.

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
- [AWS Certificate Manager Guia do usuário](#)

Trabalhe com Kinesis

Esta seção fornece exemplos de programação [Amazon Kinesis](#) usando o AWS SDK for Java 2.x.

Para obter mais informações sobre Kinesis, consulte o [Guia do Amazon Kinesis 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 do fluxo de dados Amazon Kinesis usando o método `subscribeToShard`. O Kinesis Data Streams agora emprega o recurso "fanout" aprimorado uma API de recuperação de dados HTTP/2 de baixa latência, o que facilita para desenvolvedores executar vários aplicativos de alto desempenho e baixa latência no mesmo fluxo de dados do Kinesis.

Configuração

Primeiro, crie um cliente do Kinesis 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();
```

Usar a interface do construtor

Você pode usar o método `builder` para simplificar a criação de [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 publicador, você pode usar o método `publisherTransformer` para personalizar o publicador.

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](#) no GitHub.

Usar um manipulador de respostas personalizado

Para ter o controle total do assinante e do publicador, implemente a interface `SubscribeToShardResponseHandler`.

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("Receieved initial response");
        }

        @Override
        public void onEventStream(SdkPublisher<SubscribeToShardEventStream>
publisher) {
            publisher
                // Filter to only SubscribeToShardEvents
                .filter(SubscribeToShardEvent.class)
                // Flat map into a publisher of just records
                .flatMapIterable(SubscribeToShardEvent::records)
                // Limit to 1000 total records
                .limit(1000)
                // Batch records into lists of 25
                .buffer(25)
                // Print out each record batch
                .subscribe(batch -> System.out.println("Record Batch - " +
batch));
        }

        @Override
        public void complete() {
            System.out.println("All records stream successfully");
        }

        @Override
        public void exceptionOccurred(Throwable throwable) {
            System.err.println("Error during stream - " + throwable.getMessage());
        }
    };
    return client.subscribeToShard(request, responseHandler);
}

```

Veja o [exemplo completo](#) no GitHub.

Usar 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](#) no GitHub.

Usar 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 método `subscribe`, 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](#) no GitHub.

Gravar registros de dados em um fluxo de dados Kinesis

Você pode usar o objeto [KinesisClient](#) para gravar registros de dados em um fluxo de dados do Kinesis usando o método `putRecords`. Para chamar esse método com êxito, crie um objeto [PutRecordsRequest](#). Passe o nome do fluxo de dados para o método `streamName`. Além disso, passe os dados usando o método `putRecords` (como mostrado no exemplo de código a seguir).

Importações

```
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 objeto `StockTrade` é usado como os dados que serão gravado no fluxo de dados do Kinesis. Antes de executar o exemplo, certifique-se de que você criou o fluxo de dados.

Código

```
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;

/**
 * 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 StockTradesWriter {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <streamName>

Where:
    streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
    """;

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

String streamName = args[0];
Region region = Region.US_EAST_1;
KinesisClient kinesisClient = KinesisClient.builder()
    .region(region)
    .build();

// Ensure that the Kinesis Stream is valid.
validateStream(kinesisClient, streamName);
setStockData(kinesisClient, streamName);
kinesisClient.close();
}

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](#) no GitHub.

Usar uma biblioteca de terceiros

Você pode usar outras bibliotecas de terceiros em vez de implementar um assinante personalizado. Este exemplo demonstra como usar a implementação RxJava, mas você pode usar qualquer biblioteca que implemente interfaces de streams reativos. Consulte a [página wiki do RxJava 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 usa RxJava no método do ciclo de vida `onEventStream`. 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 editor Flowable para 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](#) no GitHub.

Mais informações

- [SubscribetoShardEvent](#) na referência da API Amazon Kinesis
- [SubscribeToShard](#) na Referência da API Amazon Kinesis

Invocar, listar e excluir funções do 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 a ser passada para a Lambda função. Os nomes das funções aparecem como `arn:aws:lambda:us-east-1:123456789012:function::HelloFunction`. É possível recuperar o valor examinando a função no AWS Management Console.

Para passar dados de carga útil 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 Lambda função.

```
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 [Lambda Client](#) objeto 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 do .

```
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](#) em GitHub.

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`. É possível recuperar o valor examinando 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](#) em GitHub.

Trabalhar com o Amazon S3

Esta seção apresenta exemplos de como programar com o [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 método [forcePathStyle](#) com `true` no seu construtor de clientes para forçar o cliente a usar o endereçamento no estilo de caminho para os buckets.

O exemplo a seguir mostra um cliente de serviço configurado com uma substituição de endpoint e usando o 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();
```

Usar pontos de acesso ou pontos de acesso multirregionais

Depois que [pontos de acesso Amazon S3](#) ou [pontos de acesso multirregionais](#) forem configurados, você poderá chamar métodos de objeto, como `putObject` e `getObject`, e fornecer o identificador do ponto de acesso em vez de um nome de bucket.

Por exemplo, se o identificador ARN de um ponto de acesso for `arn:aws:s3:us-west-2:123456789012:accesspoint/test`, você poderá usar o seguinte trecho de código 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 parâmetro `bucket` pelo ARN do ponto de acesso multirregional que tenha o formato a seguir.

```
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 buckets do Amazon S3](#)
- [Trabalhar com objetos do Amazon S3](#)
- [Trabalhar com URLs pré-assinados do Amazon S3](#)
- [Acesso entre regiões para o Amazon S3](#)
- [Somos de verificação do Amazon S3 com](#)
- [Usar um cliente do S3 de alta performance: cliente do S3 baseado no AWS CRT](#)
- [Transfira arquivos e diretórios com o Gerenciador de transferências do Amazon S3](#)

Criar, listar e excluir buckets do Amazon S3

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 os buckets e suas configurações, consulte [Trabalhar com buckets do Amazon S3](#) Guia do usuário do Amazon Simple Storage Service.

Note

Melhor prática

Recomendamos habilitar a regra de ciclo de vida [AbortIncompleteMultipartUpload](#) nos buckets do 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 versionamento](#) no Guia do usuário do Amazon Simple Storage Service.

Note

Esses trechos de código pressupõem que você compreenda o material no básico e tenha configurado as credenciais padrão da AWS usando as informações contidas em [the section called “Configurar acesso de logon único para o SDK”](#).

Criar um bucket

Crie um [CreateBucketRequest](#) e forneça um nome de bucket. Passe-o para o método `createBucket` do `S3Client`. Use o `S3Client` para realizar operações adicionais, como listar ou excluir buckets, conforme mostrado nos exemplos mais adiante.

Importações

```
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`.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Faça uma solicitação de criação de bucket.

```
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;

/**
 * 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 S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String bucket = "bucket" + System.currentTimeMillis();
        System.out.println(bucket);
        createBucket(s3, bucket);
        performOperations(s3, 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](#) no GitHub.

Listar buckets

Crie um [ListBucketsRequest](#). Use o método `listBuckets` 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.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`.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .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](#) no 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 método `listObjects` 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.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.

```
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;

/**
 * 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 S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <bucket>

            Where:
                bucket - The bucket to delete (for example, bucket1).\s
            """;

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

```
String bucket = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteObjectsInBucket(s3, bucket);
s3.close();
}

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());
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucket).build();
        s3.deleteBucket(deleteBucketRequest);

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

Veja o [exemplo completo](#) no GitHub.

Excluir um bucket vazio

Crie um [DeleteBucketRequest](#) com um nome de bucket e passe-o para o método `deleteBucket` do `S3Client`.

Importações

```
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`.

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

Excluir o bucket.

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

Veja o [exemplo completo](#) no GitHub.

Trabalhar com objetos do Amazon S3

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 habilitar a regra de ciclo de vida [AbortIncompleteMultipartUpload](#) nos buckets do 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 versionamento](#) no Guia do usuário do Amazon Simple Storage Service.

Note

Esses trechos de código pressupõem que você compreenda o material no básico e tenha configurado as credenciais padrão da AWS usando as informações contidas em [the section called “Configurar acesso de logon único para o SDK”](#).

Tópicos

- [Fazer upload de um objeto](#)
- [Fazer upload de objetos em várias partes](#)
- [Excluir um objeto](#)
- [Listar objetos](#)
- [Mais exemplos](#)

Fazer upload de um objeto

Crie um [PutObjectRequest](#) e forneça um nome do bucket e nome da chave. Em seguida, use o método `putObject` do `S3Client` com um [RequestBody](#) com o conteúdo do objeto e o objeto `PutObjectRequest`. 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](#) no GitHub.

Fazer upload de objetos em várias partes

Use o método `createMultipartUpload` do `S3Client` para obter um ID de upload. Em seguida, use o método `uploadPart` para fazer upload de cada parte. Por fim, use o método `completeMultipartUpload` do `S3Client` para informar ao Amazon S3 que mescele todas as partes carregadas e conclua 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](#) no GitHub.

Excluir um objeto

Crie um [DeleteObjectRequest](#) e forneça um nome do bucket e nome da chave. Use o método `deleteObject` do `S3Client` e passe para ele o nome de um bucket e objeto a ser excluído. 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](#) no GitHub.

Copiar um objeto

Crie um [CopyObjectRequest](#) e forneça um nome de bucket no qual o objeto é copiado, um valor de string codificado por URL (consulte o método `URLEncoder.encode`) e o nome da chave do objeto. Use o método `copyObject` do `S3Client` e passe o objeto [CopyObjectRequest](#). O bucket e a chave de objeto especificados devem existir, ou o serviço retornará um erro.

Importações

```
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

```
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;

/**
 * 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 CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
bucket2).

            """;

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

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }
}
```

```
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](#) no GitHub.

Listar objetos

Crie um [ListObjectsRequest](#) e forneça o nome do bucket. Depois, invoque o método `listObjects` do `S3Client` e transmita o objeto `ListObjectsRequest`. Esse método retorna um [ListObjectsResponse](#) que contém todos os objetos do 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.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

```
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;

/**
 * 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 ListObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }
}
```

```
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.println("\n The name of the key is " + myValue.key());
            System.out.println("\n The object is " + calKb(myValue.size()) + " KBs");
            System.out.println("\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](#) no GitHub.

Mais exemplos

A seção [Exemplos de código](#) deste guia contém mais exemplos de como trabalhar com objetos do Amazon S3, incluindo como [fazer download de um objeto](#).

Trabalhar com URLs pré-assinados do Amazon S3

URLs pré-assinados concedem acesso temporário a objetos privados do S3 sem exigir que os usuários tenham credenciais ou permissões da AWS.

Por exemplo, vamos supor que Alice tenha acesso a um objeto do S3 e queira compartilhar temporariamente o acesso a esse objeto com Bob. A Alice pode gerar uma solicitação GET pré-

assinada para compartilhar com o Bob, para que ele possa baixar o objeto sem exigir acesso às credenciais da Alice. É possível gerar URLs pré-assinados para solicitações HTTP GET e HTTP PUT.

Gere um URL pré-assinado para um objeto e baixe-o (solicitação GET).

O exemplo a seguir consiste em duas partes.

- Parte 1: A Alice gera o URL pré-assinado de um objeto.
- Parte 2: O Bob baixa o objeto usando o URL pré-assinado.

Parte 1: Gerar o URL

A Alice já tem um objeto em um bucket do S3. Ela usa o código a seguir para gerar uma string de URL que o Bob pode usar em uma solicitação GET subsequente.

Importações

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
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.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest = GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL will expire
in 10 minutes.
            .getObjectRequest(objectRequest)
            .build();

        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Parte 2: Baixar o objeto

O Bob usa uma das três opções de código a seguir para baixar o objeto. Como alternativa, ele pode usar um navegador para realizar a solicitação GET.

Usar o JDK `HttpURLConnection` (desde a v1.1)

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
Capture the response body to a byte array.
```

```

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());
    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Usar o JDK `HttpClient` (desde a v11)

```

/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());
    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Usar `SdkHttpClient` do SDK para Java

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
            sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
            response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

Veja o [exemplo completo](#) e [teste](#) no GitHub.

Gerar um URL pré-assinado para um upload e, depois, fazer upload de um arquivo (solicitação PUT)

O exemplo a seguir consiste em duas partes.

- Parte 1: A Alice gera o URL pré-assinado para fazer upload de um objeto.
- Parte 2: O Bob faz upload de um arquivo usando o URL pré-assinado.

Parte 1: Gerar o URL

A Alice já tem um bucket do S3. Ela usa o código a seguir para gerar uma string de URL que o Bob pode usar em uma solicitação PUT subsequente.

Importações

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
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;
```

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

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

        PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires 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("HTTP method: [{}]", presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Parte 2: Fazer upload de um objeto de arquivo

O Bob usa uma das três opções de código a seguir para fazer upload de um arquivo.

Usar o JDK **HttpURLConnection** (desde a v1.1)

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
```

```

public void useHttpURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        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);
    }
}

```

Usar o JDK `HttpClient` (desde a v11)

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

```

```

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(new URL(presignedUrlString).toURI())
        .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
            .build(),
            HttpResponse.BodyHandlers.discarding());

    logger.info("HTTP response code is " + response.statusCode());

} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Usar `SdkHttpClient` do SDK para Java

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

```



```
try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
    HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
    logger.info("Response code: {}", response.httpResponse().statusCode());
}
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
}
```

Veja o [exemplo completo](#) e [teste](#) no GitHub.

Acesso entre regiões para o Amazon S3

Quando você trabalha com os buckets do Amazon Simple Storage Service (Amazon S3), você geralmente conhece a Região da AWS do bucket. A região com a qual você trabalha é determinada quando você cria o cliente do 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 do 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 entre diferentes regiões.

Configuração

O suporte para acesso entre regiões foi disponibilizado com a versão 2.20.111 do SDK. Use essa versão ou uma versão posterior em seu arquivo de compilação do Maven para a dependência do s3, 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 do S3, habilite o acesso entre regiões conforme mostrado no trecho. O acesso não está habilitado por padrão.

```
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 ao usar o método `putObject`, 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 futuras solicitações para o mesmo bucket, o SDK armazena em cache no cliente esse mapeamento de região.

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.

Somas de verificação do Amazon S3 com

O Amazon Simple Storage Service (Amazon S3) oferece a capacidade de especificar uma soma de verificação ao fazer upload de um objeto. Quando você especifica uma soma de verificação, ela é armazenada com o objeto e pode ser validada quando download do objeto é feito.

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 de dados confirmando que o arquivo recebido corresponde ao arquivo original. Para obter mais informações sobre o Amazon S3, consulte o [Guia do usuário do Amazon Simple Storage Service](#).

Atualmente, o Amazon S3 é compatível com quatro algoritmos de soma de verificação: SHA-1, SHA-256, CRC-32 e CRC-32C. Você tem a flexibilidade de escolher o algoritmo que melhor atende às suas necessidades e deixar que o SDK calcule a soma de verificação. Você também 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, CRC32C, SHA1 e SHA256.

O trecho de código a seguir mostra uma solicitação para fazer upload de um objeto com uma soma de verificação CRC-32. Quando o SDK envia a solicitação, ele calcula a soma de verificação CRC-32 e faz o upload do 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.

Usar 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.

Multipart Uploads

Você também pode usar somas de verificação com multipart uploads.

Fazer download de um objeto

Quando você usa o método [getObject](#) para fazer download de um objeto, o SDK valida automaticamente a soma de verificação

A solicitação no trecho a seguir direciona o SDK para validar a soma de verificação na resposta calculando a soma de verificação e comparando os valores.

Se o tiver sido feito o upload do objeto 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.

Usar um cliente do S3 de alta performance: cliente do S3 baseado no AWS CRT

O cliente do S3 da AWS baseado em CRT, construído com base no [Common Runtime \(CRT\) da AWS](#), é um cliente assíncrono alternativo do S3. 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 do S3 da AWS baseado em CRT melhora a confiabilidade da transferência caso haja uma falha na rede. A confiabilidade é aprimorada ao realizar novas tentativas de partes individuais com falha em uma transferência de arquivos sem reiniciar a transferência desde o início.

Além disso, o cliente do S3 da AWS baseado em CRT oferece um pool de conexões aprimorado e balanceamento de carga do Sistema de Nomes de Domínio (DNS), o que também melhora o throughput.

Você pode usar o cliente do S3 da AWS baseado em CRT no lugar do cliente assíncrono do S3 padrão do SDK e aproveitar imediatamente seu throughput aprimorado.

Componentes da AWS baseados em CRT no SDK

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

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

O [cliente HTTP da AWS baseado em CRT](#) é uma implementação da interface [SdkAsyncHttpClient](#) e é usado para a comunicação HTTP em geral. Essa é uma alternativa à implementação da interface `SdkAsyncHttpClient` do Netty e oferece várias vantagens.

Embora ambos os componentes usem bibliotecas do [Common Runtime da AWS](#), o cliente do S3 da AWS baseado em CRT usa a [biblioteca aws-c-s3](#) e oferece suporte aos atributos da [API de upload de várias partes do S3](#). Como o cliente HTTP da AWS baseado em CRT é destinado ao uso geral, ele não oferece suporte aos atributos da API de upload de várias partes do S3.

Adicionar dependências para usar o cliente do S3 da AWS baseado em CRT

Para usar o cliente do S3 da 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. Pesquisar 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>
```

Criar uma instância do cliente do S3 da AWS baseado em CRT

Crie uma instância do cliente do S3 da 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 criador de cliente CRT da AWS. Você pode alternar do cliente assíncrono do S3 padrão para o cliente da AWS baseado em CRT alterando o método de construtor.

```
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 CRT da AWS. Obtenha o construtor padrão chamando `S3AsyncClient#builder()`.

Usar o cliente do S3 baseado em CRT da AWS

Use o cliente do S3 da AWS baseado em CRT para chamar as operações de API do Amazon S3. O exemplo a seguir demonstra as operações [PutObject](#) e [GetObject](#) 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>),
        AsyncResponseTransformerToFile(Paths.get(<FILE_NAME>)))
        .join();
```

Transfira arquivos e diretórios com o Gerenciador de transferências do Amazon S3

O gerenciador de transferências do S3 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 desenvolvido com base no [cliente do S3 com base no AWS CRT](#), o gerenciador de transferências do S3 pode aproveitar as melhorias de performance, como a [API de multipart upload](#) e [buscas por intervalo de bytes](#).

Com o gerenciador de transferências do S3, você também pode monitorar o progresso de uma transferência em tempo real e pausar a transferência para execução posterior.

Conceitos básicos

Adicionar 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.

- Usar a versão **2.19.1** ou superior do SDK para Java 2.x.
- Adicionar o artefato `s3-transfer-manager` como uma dependência.
- Adicionar o artefato `aws-crt` como uma dependência na versão **0.20.3** ou superior.

O exemplo de código a seguir mostra como configurar as dependências do 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>
</project>
```

```
</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](#).

Criar uma instância do gerenciador de transferências do S3

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 gerenciador de transferências do S3 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 dependência `aws-crt` não estiver incluída no arquivo de compilação, o gerenciador de transferências do S3 será criado com base no cliente assíncrono do S3 padrão usado no SDK para Java 2.x.

Carregar um arquivo em um bucket do S3

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.

Para carregar um arquivo para o Amazon S3 usando o gerenciador de transferências do S3, passe um objeto [UploadFileRequest](#) para o método [uploadFile](#) do `S3TransferManager`.

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.

```
public String uploadFile(S3TransferManager transferManager, String bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

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.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

Baixar um arquivo de um bucket do S3

O exemplo a seguir mostra um exemplo de download junto com o uso opcional de um [LoggingTransferListener](#), que registra o progresso do download.

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.

```
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();
}
```

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.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

Adicionar um objeto a um bucket do Amazon S3

O exemplo a seguir mostra como copiar um objeto com o gerenciador de transferência do S3.

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 objeto `Copy` retornado pelo método `copy` do `S3TransferManager` representa o processo de cópia. Depois que o processo de cópia for concluído, o [CompletedCopy](#) objeto conterá detalhes sobre a resposta.

```
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

Para realizar uma cópia entre regiões com o S3 Transfer Manager, habilite `crossRegionAccessEnabled` no construtor de clientes S3 AWS baseado em CRT, conforme mostrado no trecho a seguir.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

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;
```

Carregamento de um diretório local para um bucket do S3

O exemplo a seguir demonstra como fazer upload de um diretório local para o 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 objeto `CompletedDirectoryUpload` contém informações sobre os resultados da transferência, incluindo quais arquivos falharam na transferência.

```
public Integer uploadDirectory(S3TransferManager transferManager,
```

```
URI 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();
}
```

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.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

Baixar objetos do bucket do S3 para um diretório local

É possível baixar os objetos em um bucket do S3 para um diretório local, conforme mostrado no exemplo a seguir.

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 objeto

`CompletedDirectoryDownload` contém informações sobre os resultados da transferência, incluindo quais arquivos falharam na transferência.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

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.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;
```

Ver exemplos completos

[GitHub contém o código completo](#) de todos os exemplos desta página.

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 compile um objeto [CreateTopicRequest](#) com o nome do conjunto de tópicos usando o método `name()` no compilador. Envie o objeto de solicitação para o Amazon SNS usando o método `createTopic()` do [SnsClient](#). É possível capturar o resultado dessa solicitação como um objeto [CreateTopicResponse](#), 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](#) no GitHub.

Listar seus tópicos do Amazon SNS

Para recuperar uma lista dos tópicos existentes do Amazon SNS, compile um objeto [ListTopicsRequest](#). Envie o objeto de solicitação para o Amazon SNS usando o método `listTopics()` do `SnsClient`. É possível capturar o resultado dessa solicitação como um objeto [ListTopicsResponse](#).

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](#) no 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 iniciar, compile um objeto [SubscribeRequest](#). Especifique o canal de comunicação (por exemplo, lambda ou email) como `protocol()`. Defina o `endpoint()` como o local de saída relevante (por exemplo, o ARN de uma função do Lambda ou um endereço de e-mail) e defina o ARN do tópico ao qual você deseja se inscrever como `topicArn()`. Envie o objeto de solicitação para o Amazon SNS usando o método `subscribe()` do `SnsClient`. É possível capturar o resultado dessa solicitação como um objeto [SubscribeResponse](#).

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](#) no 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 iniciar, compile um objeto [PublishRequest](#). 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`. É possível capturar o resultado dessa solicitação como um objeto [PublishResponse](#).

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](#) no 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, compile um objeto [UnsubscribeRequest](#) e defina o ARN do tópico do qual você deseja cancelar a inscrição como o `subscriptionArn()`. Envie o objeto de solicitação para o SNS usando o método `unsubscribe()` do `SnsClient`. É possível capturar o resultado dessa solicitação como um objeto [UnsubscribeResponse](#).

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](#) no GitHub.

Excluir um tópico

Para excluir um tópico do Amazon SNS, primeiro compile um objeto [DeleteTopicRequest](#) com o ARN do tópico definido como o método `topicArn()` no compilador. Envie o objeto de solicitação para o Amazon SNS usando o método `deleteTopic()` do `SnsClient`. É possível capturar o resultado dessa solicitação como um objeto [DeleteTopicResponse](#), 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](#) no GitHub.

Para obter mais informações, consulte o [Guia do desenvolvedor do Amazon Simple Notification Service](#).

Trabalhe 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

- [Trabalhar com filas de mensagens do Amazon Simple Queue Service](#)
- [Enviar, receber e excluir mensagens do Amazon Simple Queue Service](#)

Trabalhar com filas de mensagens do Amazon Simple Queue Service

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 as filas e as diferenças entre esses tipos, consulte o [Guia do desenvolvedor do Amazon Simple Queue Service](#).

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.

A variável `sqsClient` usada nos exemplos a seguir pode ser criada a partir do trecho de código a seguir.

```
SqsClient sqsClient = SqsClient.create();
```

Quando você cria um `SqsClient` usando o `create()` método estático, o SDK configura a região usando a cadeia de [provedores da região padrão e as credenciais usando a cadeia](#) de provedores de [credenciais padrão](#).

Criar uma fila

Use o `SqsClient`'s `createQueue` método e forneça um [CreateQueueRequest](#) objeto que descreva os parâmetros da fila, conforme mostrado no trecho de código a seguir.

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();

        sqsClient.createQueue(createQueueRequest);
```

Veja a [amostra completa](#) em GitHub.

Listar filas

Para listar as Amazon Simple Queue Service filas da sua conta, chame o `SqsClient`'s `listQueues` método com um [ListQueuesRequest](#) objeto.

Quando você usa a forma do [listQueues](#) método que não usa parâmetros, o serviço retorna todas as filas — até 1.000 filas.

Você pode fornecer um prefixo de nome de fila ao [ListQueuesRequest](#) objeto para limitar os resultados às filas que correspondam a esse prefixo, conforme mostrado no código a seguir.

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 a [amostra completa](#) em GitHub.

Obter o URL de uma fila

O código a seguir mostra como obter o URL de uma fila chamando 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;
```

Veja a [amostra completa](#) em GitHub.

Excluir uma fila

Forneça o [URL](#) da fila para o [DeleteQueueRequest](#) objeto. Em seguida, chame o `SqsClient`'s `deleteQueue` método para excluir uma fila, conforme mostrado no código a seguir.

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 a [amostra completa](#) em GitHub.

Mais informações

- [CreateQueue](#) na Referência da Amazon Simple Queue Service API
- [GetQueueUrl](#) na Referência da Amazon Simple Queue Service API

- [ListQueues](#) na Referência da Amazon Simple Queue Service API
- [DeleteQueue](#) na Referência da Amazon Simple Queue Service API

Enviar, receber e excluir mensagens do Amazon Simple Queue Service

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](#).

A variável `sqsClient` usada nos exemplos a seguir pode ser criada a partir do trecho de código a seguir.

```
SqsClient sqsClient = SqsClient.create();
```

Quando você cria um `SqsClient` usando o `create()` método estático, o SDK configura a região usando a cadeia de [provedores da região padrão e as credenciais usando a cadeia](#) de provedores de [credenciais padrão](#).

Enviar uma mensagem

Adicione uma única mensagem a uma Amazon Simple Queue Service fila chamando o `sendMessage` método `SqsClient` cliente. Forneça um [SendMessageRequest](#) objeto que contenha a [URL](#) da fila, o corpo da mensagem e o 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());

sqsClient.sendMessage(sendMsgRequest);
```

Enviar várias mensagens em uma solicitação

Envie mais de uma mensagem em uma única solicitação usando o método do `sendMessageBatch` do `SqsClient`. Esse método usa 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();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

Veja a [amostra completa](#) em GitHub.

Recuperar mensagens

Recupere todas as mensagens que estejam atualmente na fila chamando o método do `receiveMessage` do `SqsClient`. Esse método usa 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

```
    try {
        ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
            .queueUrl(queueUrl)
            .numberOfMessages(5)
            .build();
        List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
        return messages;
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
```

Veja a [amostra completa](#) em GitHub.

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`'s [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();
    sqsClient.deleteMessage(deleteMessageRequest);
}
```

Veja a [amostra completa](#) em GitHub.

Mais informações

- [Como as filas do Amazon Simple Queue Service funcionam](#) no Guia do Desenvolvedor do Amazon Simple Queue Service
- [SendMessage](#) na Referência da Amazon Simple Queue Service API
- [SendMessageBatch](#) na Referência da Amazon Simple Queue Service API
- [ReceiveMessage](#) na Referência da Amazon Simple Queue Service API
- [DeleteMessage](#) na Referência da Amazon Simple Queue Service 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 transcrito em tempo real.

Consulte [Transcrição de streaming](#) no Guia do desenvolvedor do Amazon Transcribe para saber mais sobre esse recurso.

Consulte [Conceitos básicos](#) no Guia do desenvolvedor do Amazon Transcribe para começar a usar o 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](#) no GitHub.

Criar um publicador

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 final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;

        private SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                        }
                    } while (demand.get() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }
    }
}
```

```
                break;
            }
        } while (demand.decrementAndGet() > 0);
    } catch (TranscribeStreamingException e) {
        subscriber.onError(e);
    }
});
}

@Override
public void cancel() {

}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer;
    byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

    int len = 0;
    try {
        len = inputStream.read(audioBytes);

        if (len <= 0) {
            audioBuffer = ByteBuffer.allocate(0);
        } else {
            audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
        }
    } catch (IOException e) {
        throw new UncheckedIOException(e);
    }

    return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
}
```

Veja o [exemplo completo](#) no GitHub.

Criar o cliente e iniciar o streaming

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 do Amazon Transcribe.

Depois, use o método `startStreamTranscription` de `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.StartStreamTranscriptionResponseHandler;
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](#) no GitHub.

Mais informações

- [Como funciona](#) no Guia do desenvolvedor do Amazon Transcribe.
- [Conceitos básicos do streaming de áudio](#) no Guia do desenvolvedor do Amazon Transcribe.

Exemplos de código do SDK para Java 2.x

Os exemplos de código neste tópico mostram como usar o AWS SDK for Java 2.x with 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 para Java 2.x](#)
- [Exemplos de serviço cruzado usando o SDK para Java 2.x](#)

Ações e cenários usando o SDK para 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 Serviços 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 para Java 2.x](#)
- [Exemplos de Application Auto Scaling usando o SDK for Java 2.x](#)
- [Exemplos do Application Recovery Controller usando o SDK para Java 2.x](#)
- [Exemplos de Aurora usando o SDK para Java 2.x](#)
- [Exemplos de Auto Scaling usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Bedrock usando o SDK para Java 2.x](#)

- [Exemplos de Amazon Bedrock Runtime usando o SDK para 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 do Amazon Cognito Identity usando o SDK para Java 2.x](#)
- [Exemplos do provedor de identidade do Amazon Cognito usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Comprehend usando o SDK para Java 2.x](#)
- [Exemplos do DynamoDB usando o SDK para Java 2.x](#)
- [Exemplos do Amazon EC2 usando o SDK para Java 2.x](#)
- [Exemplos do Amazon ECS usando o SDK para Java 2.x](#)
- [Exemplos do Elastic Load Balancing usando o SDK for Java 2.x](#)
- [MediaStore exemplos 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 do Forecast usando o SDK para Java 2.x](#)
- [AWS Glue exemplos usando o SDK for Java 2.x](#)
- [HealthImaging exemplos usando o SDK for Java 2.x](#)
- [Exemplos do IAM usando o SDK para Java 2.x](#)
- [AWS IoT exemplos usando o SDK for Java 2.x](#)
- [AWS IoT data exemplos usando o SDK for Java 2.x](#)
- [Exemplos do Amazon Keyspaces usando o SDK para Java 2.x](#)
- [Exemplos do Kinesis usando o SDK para Java 2.x](#)
- [AWS KMS exemplos usando o SDK for Java 2.x](#)
- [Exemplos do Lambda usando o SDK para Java 2.x](#)
- [MediaConvert exemplos usando o SDK for Java 2.x](#)
- [Exemplos do Migration Hub usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Personalize usando o SDK para Java 2.x](#)
- [Exemplos de eventos do Amazon Personalize usando o SDK for Java 2.x](#)

- [Exemplos do Amazon Personalize Runtime usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Pinpoint usando o SDK para Java 2.x](#)
- [Exemplos da API SMS and Voice do Amazon Pinpoint usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Polly usando o SDK para Java 2.x](#)
- [Exemplos do Amazon RDS usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Redshift usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Rekognition usando o SDK for Java 2.x](#)
- [Exemplos de registro de domínios do Route 53 usando o SDK para Java 2.x](#)
- [Exemplos do Amazon S3 usando o SDK para Java 2.x](#)
- [Exemplos do S3 Glacier usando o SDK para Java 2.x](#)
- [SageMaker exemplos usando o SDK for Java 2.x](#)
- [Exemplos do Secrets Manager usando o SDK para Java 2.x](#)
- [Exemplos do Amazon SES usando o SDK para Java 2.x](#)
- [Exemplos de API v2 do Amazon SES usando o SDK para Java 2.x](#)
- [Exemplos do Amazon SNS usando o SDK para Java 2.x](#)
- [Exemplos do Amazon SQS usando o SDK para Java 2.x](#)
- [Exemplos do Step Functions usando o SDK para 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 para Java 2.x](#)
- [Exemplos do Amazon Textract usando o SDK para Java 2.x](#)
- [Exemplos do Amazon Transcribe usando o SDK para Java 2.x](#)

Exemplos de API Gateway usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 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 [AWS Code Examples Repository](#).

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

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

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

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

```
        return "";  
    }
```

- 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 [AWS Code Examples Repository](#).

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

- 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 [AWS Code Examples Repository](#).

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

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

        apiGateway.deleteDeployment(request);
        System.out.println("Deployment was deleted");


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

- Para obter detalhes da API, consulte [DeleteDeployment](#) a 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 [AWS Code Examples Repository](#).

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

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

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

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

- Para obter detalhes da API, consulte [CreateDeployment](#) Referência AWS SDK for Java 2.x da API.

Exemplos de Application 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 with Application 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Desativa um recurso

O exemplo de código a seguir mostra como desativar um recurso do Application Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
```

```
* 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 DisableDynamoDBAutoscaling {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <tableId> <policyName>\s  
  
            Where:  
            tableId - The table Id value (for example, table/Music).\s  
            policyName - The name of the policy (for example, $Music5-scaling-  
policy).  
  
            "";  
        if (args.length != 2) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        ApplicationAutoScalingClient appAutoScalingClient =  
ApplicationAutoScalingClient.builder()  
            .region(Region.US_EAST_1)  
            .build();  
  
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;  
        ScalableDimension tableWCUs =  
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;  
        String tableId = args[0];  
        String policyName = args[1];  
  
        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);  
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);  
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);  
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);  
    }  
}
```

```
public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
    try {
        DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
        .policyName(policyName)
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceId(tableId)
        .build();

        appAutoScalingClient.deleteScalingPolicy(delSPRequest);
        System.out.println(policyName + " was deleted successfully.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the scaling policy was deleted
public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
    .scalableDimension(tableWCUs)
    .serviceNamespace(ns)
    .resourceId(tableId)
    .build();

    DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    try {
        DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
        .scalableDimension(tableWCUs)
```

```
        .serviceNamespace(ns)
        .resourceId(tableId)
        .build();

        appAutoScalingClient.deregisterScalableTarget(targetRequest);
        System.out.println("The scalable target was deregistered.");

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}
}
```

- Para obter detalhes da API, consulte [DeleteScalingPolicy](#) a Referência AWS SDK for Java 2.x da API.

Registra um recurso

O exemplo de código a seguir mostra como registrar um recurso do Application Auto Scaling.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
    software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
    software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyConfiguration;
import java.util.List;

/**
 * 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 EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableId> <roleARN> <policyName>\s

            Where:
                tableId - The table Id value (for example, table/Music).
                roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
                policyName - The name of the policy to create.

            """;

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

        System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }
}
```

```
public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
    try {
        RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
            .serviceNamespace(ns)
            .scalableDimension(tableWCUs)
            .resourceId(tableId)
            .roleARN(roleARN)
            .minCapacity(5)
            .maxCapacity(10)
            .build();

        appAutoScalingClient.registerScalableTarget(targetRequest);
        System.out.println("You have registered " + tableId);

    } catch (ApplicationAutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Verify that the target was created.
public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
    DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
        .scalableDimension(tableWCUs)
        .serviceNamespace(ns)
        .resourceIds(tableId)
        .build();

    DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
    System.out.println("DescribeScalableTargets result: ");
    System.out.println(response);
}

// Configure a scaling policy.
public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
    // Check if the policy exists before creating a new one.
```

```
DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
    .serviceNamespace(ns)
    .resourceId(tableId)
    .scalableDimension(tableWCUs)
    .build());

if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
    // If policies exist, consider updating an existing policy instead of
creating a new one.
    System.out.println("Policy already exists. Consider updating it
instead.");
    List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
    for (ScalingPolicy pol : polList) {
        System.out.println("Policy name:" +pol.policyName());
    }
} else {
    // If no policies exist, proceed with creating a new policy.
    PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
    .build();

    TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
    .predefinedMetricSpecification(specification)
    .targetValue(50.0)
    .scaleInCooldown(60)
    .scaleOutCooldown(60)
    .build();

    PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
    .targetTrackingScalingPolicyConfiguration(policyConfiguration)
    .serviceNamespace(ns)
    .scalableDimension(tableWCUs)
    .resourceId(tableId)
    .policyName(policyName)
    .policyType(PolicyType.TARGET_TRACKING_SCALING)
    .build();

    try {
```



```
        appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
        System.out.println("You have successfully created a scaling policy
for an Application Auto Scaling scalable target");
    } catch (ApplicationAutoScalingException e) {
        System.err.println("Error: " + e.awsErrorDetails().errorMessage());
    }
}
}
```

- Para obter detalhes da API, consulte [RegisterScalableTarget](#) Referência AWS SDK for Java 2.x da API.

Exemplos do Application Recovery Controller usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Obter o estado de um controle de roteamento

O exemplo de código a seguir mostra como obter o estado do 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 [AWS Code Examples Repository](#).

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn) {
    // As a best practice, we recommend choosing a random cluster endpoint to
    get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
    practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- Para obter detalhes da API, consulte [GetRoutingControlState](#) a 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 [AWS Code Examples Repository](#).

```

    public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
    String routingControlArn,
    String routingControlState) {
    // As a best practice, we recommend choosing a random cluster endpoint to
get or
    // set routing control states.
    // For more information, see
    // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
    Collections.shuffle(clusterEndpoints);
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()

                    .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}

```

- Para obter detalhes da API, consulte [UpdateRoutingControlState](#) a Referência AWS SDK for Java 2.x da API.

Exemplos de Aurora usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 Aurora

Os exemplos de código a seguir mostram como começar a usar o Aurora.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
```

```
Region region = Region.US_EAST_1;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

describeClusters(rdsClient);
rdsClient.close();
}

public static void describeClusters(RdsClient rdsClient) {
    DescribeDBClustersIterable clustersIterable =
rdsClient.describeDBClustersPaginator();
    clustersIterable.stream()
        .flatMap(r -> r.dbClusters().stream())
        .forEach(cluster -> System.out
            .println("Database name: " + cluster.databaseName() + " Arn
= " + cluster.dbClusterArn()));
}
}
```

- Para obter detalhes da API, consulte [DescribeDBClusters](#) na Referência da API do AWS SDK for Java 2.x .

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um cluster de banco de dados

O exemplo de código a seguir mostra como criar um cluster do Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

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


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

- Para obter detalhes da API, consulte [CreateDBCluster](#) na Referência da 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 do Aurora DB.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

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


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

- 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 de cluster do Aurora DB.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

```
try {
    CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

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

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

- 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 do DB em um cluster do Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```



```

        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build();

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

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

```

- Para ter detalhes da API, consulte [CreateDBInstance](#) na Referência da 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 do Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

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

```

```

        .skipFinalSnapshot(true)
        .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier + " was deleted!");
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

```

- Para ter detalhes da API, consulte [DeleteDBCluster](#) na Referência da 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 de cluster do Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

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

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();

```

```

        int listSize = instanceList.size();
        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 do Aurora DB.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

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

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

- Para ter detalhes da API, consulte [DeleteDBInstance](#) na Referência da 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 grupos de parâmetros de cluster do Aurora DB.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

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

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

- 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 snapshots de cluster do Aurora DB.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

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

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

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

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

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

- 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 clusters do Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
```

```

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

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

```

- Para obter detalhes da API, consulte [DescribeDBClusters](#) na Referência da API do AWS SDK for Java 2.x .

Descrever instâncias de banco de dados

O exemplo de código a seguir mostra como descrever instâncias do Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
// Waits until the database instance is available.
```



```
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();


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

- Para ter detalhes da API, consulte [DescribeDBInstances](#) na Referência da 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 de mecanismo de banco de dados do Aurora.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

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

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

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

- 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 Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

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

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


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

- 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 de cluster do Aurora DB.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

        DescribeDbClusterParametersResponse response = rdsClient
            .describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
        }
    }
}
```

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

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

- 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 os parâmetros de um grupo de parâmetros de cluster do Aurora DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
```

```
DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .maxRecords(20)
    .build();

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

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

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

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

```

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

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

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

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

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

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

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

    } catch (RdsException e) {
```

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

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

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

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

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

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

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

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

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

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

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                }
            }
        }
    }
}
```

```

        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

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

public static String createDBInstanceCluster(RdsClient rdsClient,
    String dbInstanceIdentifier,
    String dbInstanceClusterIdentifier,
    String instanceClass) {
    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .engine("aurora-mysql")
            .dbInstanceClass(instanceClass)
            .build();

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

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

public static String getListInstanceClasses(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()

```



```

        .engine("aurora-mysql")
        .maxRecords(20)
        .build();

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

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

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

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {

```

```
        instanceReady = true;
    } else {
        System.out.print(".");
        Thread.sleep(sleepTime * 1000);
    }
}
}
System.out.println("Database cluster is available!");

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

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

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

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

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
```

```

        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("aurora-mysql")
    .build();

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

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

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

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

        ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
        System.out.println(

```

```

        "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

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

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

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

```

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

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

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

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

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

public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
```

```
        .description("Created by using the AWS SDK for Java")
        .build();

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

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

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

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

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

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da 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 para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * 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
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
```



```
        System.out.println("Group Name: " + group.autoScalingGroupName());
        System.out.println("Group ARN: " + group.autoScalingGroupARN());
    });
}
}
```

- Para obter detalhes da API, consulte [DescribeAutoScalingGroups](#) na Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um grupo

O exemplo de código a seguir mostra como criar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
```

```
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * 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
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
                """;

        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)
            .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
    software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * 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
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

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

        String groupName = args[0];
```

```
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    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.

Desabilitar coleção 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 [AWS Code Examples Repository](#).

```
public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");


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

- Para obter detalhes da API, consulte [DisableMetricsCollection](#) na Referência AWS SDK for Java 2.x da API.

Habilitar coleção 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 [AWS Code Examples Repository](#).

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [EnableMetricsCollection](#) na Referência AWS SDK for Java 2.x da API.

Obter informações sobre grupos

O exemplo de código a seguir mostra como obter informação sobre grupos do Auto Scaling.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
    software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * 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
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <groupName>

            Where:
                groupName - The name of the Auto Scaling group.
            """;

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



```
String groupName = args[0];
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

String instanceId = getAutoScaling(autoScalingClient, groupName);
System.out.println(instanceId);
autoScalingClient.close();
}

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](#) na Referência AWS SDK for Java 2.x da API.

Obter informação sobre as 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 [AWS Code Examples Repository](#).

```
public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
        .builder()
        .instanceIds(id)
        .build();

        DescribeAutoScalingInstancesResponse response = autoScalingClient
        .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance : instances) {
            System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
        }

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

- Para obter detalhes da API, consulte [DescribeAutoScalingInstances](#) a Referência AWS SDK for Java 2.x da API.

Obter informações sobre as atividades de escalabilidade

O exemplo de código a seguir mostra como obter informação sobre 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 [AWS Code Examples Repository](#).

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response = autoScalingClient
            .describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity : activities) {
            System.out.println("The activity Id is " + activity.activityId());
            System.out.println("The activity details are " +
activity.details());
        }

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

- Para obter detalhes da API, consulte [DescribeScalingActivities](#) a Referência AWS SDK for Java 2.x da API.

Definir 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 [AWS Code Examples Repository](#).

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
        .autoScalingGroupName(groupName)
        .desiredCapacity(2)
        .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

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

- 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 [AWS Code Examples Repository](#).

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
        TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance " + instanceId);

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

- Para obter detalhes da API, consulte [TerminateInstanceInAutoScalingGroup](#) 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 [AWS Code Examples Repository](#).

```
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 [UpdateAutoScalingGroup](#) 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 saúde atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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.deleteAutoScalingGroup(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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
    if (!wasSuccessul) {
        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 terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    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.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
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 da API do AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)

- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Gerenciar grupos e instâncias

O código de exemplo a seguir mostra como:

- Crie um grupo do Amazon EC2 Auto Scaling com um modelo de execução 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 ajuste de escala que ocorrem em resposta às solicitações do usuário e às alterações 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 = ""

            Usage:
                <groupName> <launchTemplateName> <vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
                vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
            """;

        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)
            .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,
groupNames);
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.numberOfWorkingAutoScalingGroups());

        } 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 da API do AWS SDK for Java 2.x .
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Exemplos do Amazon Bedrock usando o SDK para 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 Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Obter detalhes sobre um modelo de base do Amazon Bedrock

O exemplo de código a seguir mostra como obter detalhes de um modelo de base do Amazon Bedrock.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obtenha detalhes sobre um modelo básico usando o cliente síncrono Amazon Bedrock.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
```

```

    * @return An object containing the foundation model's details.
    */
    public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
        try {
            GetFoundationModelResponse response = bedrockClient.getFoundationModel(
                r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = response.modelDetails();

            System.out.println(" Model ID:                " + model.modelId());
            System.out.println(" Model ARN:                " +
model.modelArn());
            System.out.println(" Model Name:                " +
model.modelName());
            System.out.println(" Provider Name:            " +
model.providerName());
            System.out.println(" Lifecycle status:         " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:         " +
model.inputModalities());
            System.out.println(" Output modalities:        " +
model.outputModalities());
            System.out.println(" Supported customizations: " +
model.customizationsSupported());
            System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ValidationException e) {
            throw new IllegalArgumentException(e.getMessage());
        } catch (SdkException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
}

```

Obtenha detalhes sobre um modelo básico usando o cliente assíncrono Amazon Bedrock.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
 * @return An object containing the foundation model's details.
 */
public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
    try {
        CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
            r -> r.modelIdentifier(modelIdentifier)
        );

        FoundationModelDetails model = future.get().modelDetails();

        System.out.println(" Model ID:                " + model.modelId());
        System.out.println(" Model ARN:                " +
model.modelArn());
        System.out.println(" Model Name:                " +
model.modelName());
        System.out.println(" Provider Name:            " +
model.providerName());
        System.out.println(" Lifecycle status:        " +
model.modelLifecycle().statusAsString());
        System.out.println(" Input modalities:        " +
model.inputModalities());
        System.out.println(" Output modalities:        " +
model.outputModalities());
        System.out.println(" Supported customizations: " +
model.customizationsSupported());
        System.out.println(" Supported inference types: " +
model.inferenceTypesSupported());
        System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

        return model;
    } catch (ExecutionException e) {
        if (e.getMessage().contains("ValidationException")) {
            throw new IllegalArgumentException(e.getMessage());
        } else {
```



```

        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
    throw new RuntimeException(e);
}
}
}

```

- Para obter detalhes da API, consulte [GetFoundationModel](#) na Referência AWS SDK for Java 2.x da API.

Listar modelos de base Amazon Bedrock disponíveis

O exemplo de código a seguir mostra como listar modelos de base disponíveis do Amazon Bedrock.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Liste os modelos de fundação Amazon Bedrock disponíveis usando o cliente síncrono Amazon Bedrock.

```

/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary> listFoundationModels(BedrockClient
bedrockClient) {

    try {
        ListFoundationModelsResponse response =
bedrockClient.listFoundationModels(r -> {});
    }
}

```

```

        List<FoundationModelSummary> models = response.modelSummaries();

        if (models.isEmpty()) {
            System.out.println("No available foundation models in " +
region.toString());
        } else {
            for (FoundationModelSummary model : models) {
                System.out.println("Model ID: " + model.modelId());
                System.out.println("Provider: " + model.providerName());
                System.out.println("Name:      " + model.modelName());
                System.out.println();
            }
        }

        return models;

    } catch (SdkClientException e) {
        System.err.println(e.getMessage());
        throw new RuntimeException(e);
    }
}

```

Liste os modelos de fundação Amazon Bedrock disponíveis usando o cliente assíncrono Amazon Bedrock.

```

/**
 * Lists Amazon Bedrock foundation models that you can use.
 * You can filter the results with the request parameters.
 *
 * @param bedrockClient The async service client for accessing Amazon Bedrock.
 * @return A list of objects containing the foundation models' details
 */
public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
    try {
        CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

        List<FoundationModelSummary> models = future.get().modelSummaries();

        if (models.isEmpty()) {

```

```
        System.out.println("No available foundation models in " +
region.toString());
    } else {
        for (FoundationModelSummary model : models) {
            System.out.println("Model ID: " + model.modelId());
            System.out.println("Provider: " + model.providerName());
            System.out.println("Name:      " + model.modelName());
            System.out.println();
        }
    }

    return models;

} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
    throw new RuntimeException(e);
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
    throw new RuntimeException(e);
}
}
```

- Para obter detalhes da API, consulte [ListFoundationModels](#) na Referência AWS SDK for Java 2.x da API.

Exemplos de Amazon Bedrock Runtime usando o SDK para 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 Bedrock 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Geração de imagens com o Amazon Titan Image Generator G1

O exemplo de código a seguir mostra como invocar o modelo Amazon Titan Image Generator no Amazon Bedrock para geração de imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque o modelo Amazon Titan Image Generator G1 de forma assíncrona para gerar imagens.

```
/**
 * Invokes the Amazon Titan image generation model to create an image using the
 * input
 * provided in the request body.
 *
 * @param prompt The prompt that you want Amazon Titan to use for image
 *               generation.
 * @param seed   The random noise seed for image generation (Range: 0 to
 *               2147483647).
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeTitanImage(String prompt, long seed) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and default values for Titan Image models refer
     to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     titan-
     * image.html
     */
}
```

```
    */
    String titanImageModelId = "amazon.titan-image-generator-v1";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    var textToImageParams = new JSONObject().put("text", prompt);

    var imageGenerationConfig = new JSONObject()
        .put("numberOfImages", 1)
        .put("quality", "standard")
        .put("cfgScale", 8.0)
        .put("height", 512)
        .put("width", 512)
        .put("seed", seed);

    JSONObject payload = new JSONObject()
        .put("taskType", "TEXT_IMAGE")
        .put("textToImageParams", textToImageParams)
        .put("imageGenerationConfig", imageGenerationConfig);

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload.toString()))
        .modelId(titanImageModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                System.out.println("Model invocation failed: " + exception);
            }
        });

    String base64ImageData = "";
    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        base64ImageData = responseBody
```

```

        .getJSONArray("images")
        .getString(0);

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return base64ImageData;
}

```

Invoke o modelo Amazon Titan Image Generator G1 para gerar imagens.

```

/**
 * Invokes the Amazon Titan image generation model to create an image using
the
 * input
 * provided in the request body.
 *
 * @param prompt The prompt that you want Amazon Titan to use for image
 * generation.
 * @param seed The random noise seed for image generation (Range: 0 to
 * 2147483647).
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeTitanImage(String prompt, long seed) {
    /**
 * The different model providers have individual request and
response formats.
 * For the format, ranges, and default values for Titan Image models
refer to:
 * https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-titan-
 * image.html
 */
    String titanImageModelId = "amazon.titan-image-generator-v1";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

```

```
.credentialsProvider(ProfileCredentialsProvider.create())
    .build();

var textToImageParams = new JSONObject().put("text", prompt);

var imageGenerationConfig = new JSONObject()
    .put("numberOfImages", 1)
    .put("quality", "standard")
    .put("cfgScale", 8.0)
    .put("height", 512)
    .put("width", 512)
    .put("seed", seed);

JSONObject payload = new JSONObject()
    .put("taskType", "TEXT_IMAGE")
    .put("textToImageParams", textToImageParams)
    .put("imageGenerationConfig",
imageGenerationConfig);

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload.toString()))
    .modelId(titanImageModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

InvokeModelResponse response = client.invokeModel(request);

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

String base64ImageData = responseBody
    .getJSONArray("images")
    .getString(0);

return base64ImageData;
}
```

- Para obter detalhes da API, consulte [InvokeModel](#) na Referência AWS SDK for Java 2.x da API.

Geração de imagens com o Stable Diffusion XL da Stability.ai

O exemplo de código a seguir mostra como invocar o modelo Stable Diffusion XL da Stability.ai no Amazon Bedrock para a geração de imagens.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque o modelo de base Stable Diffusion XL da Stability.ai de forma assíncrona para gerar imagens.

```
/**
 * Asynchronously invokes the Stability.ai Stable Diffusion XL model to create
 * an image based on the provided input.
 *
 * @param prompt      The prompt that guides the Stable Diffusion model.
 * @param seed        The random noise seed for image generation (use 0 or omit
 *                    for a random seed).
 * @param stylePreset The style preset to guide the image model towards a
 *                    specific style.
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeStableDiffusion(String prompt, long seed, String
stylePreset) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and available style_presets of Stable Diffusion
     * models refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     stability-diffusion.html
     */

    String stableDiffusionModelId = "stability.stable-diffusion-xl";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
```



```
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    JSONArray wrappedPrompt = new JSONArray().put(new JSONObject().put("text",
prompt));
    JSONObject payload = new JSONObject()
        .put("text_prompts", wrappedPrompt)
        .put("seed", seed);

    if (stylePreset != null && !stylePreset.isEmpty()) {
        payload.put("style_preset", stylePreset);
    }

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload.toString()))
        .modelId(stableDiffusionModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                System.out.println("Model invocation failed: " + exception);
            }
        });

    String base64ImageData = "";
    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        base64ImageData = responseBody
            .getJSONArray("artifacts")
            .getJSONObject(0)
            .getString("base64");
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }
}
```

```

        return base64ImageData;
    }

```

Invoque o modelo de base Stable Diffusion XL da Stability.ai para gerar imagens.

```

/**
 * Invokes the Stability.ai Stable Diffusion XL model to create an image
based
 * on the provided input.
 *
 * @param prompt      The prompt that guides the Stable Diffusion model.
 * @param seed        The random noise seed for image generation (use 0 or
omit
 *                    for a random seed).
 * @param stylePreset The style preset to guide the image model towards a
 *                    specific style.
 * @return A Base64-encoded string representing the generated image.
 */
public static String invokeStableDiffusion(String prompt, long seed, String
stylePreset) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and available style_presets of Stable
Diffusion
     * models refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-stability-diffusion.html
     */

    String stableDiffusionModelId = "stability.stable-diffusion-xl";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    JSONArray wrappedPrompt = new JSONArray().put(new
JSONObject().put("text", prompt));

```

```
JSONObject payload = new JSONObject()
    .put("text_prompts", wrappedPrompt)
    .put("seed", seed);

if (!(stylePreset == null || stylePreset.isEmpty())) {
    payload.put("style_preset", stylePreset);
}

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload.toString()))
    .modelId(stableDiffusionModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

InvokeModelResponse response = client.invokeModel(request);

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

String base64ImageData = responseBody
    .getJSONArray("artifacts")
    .getJSONObject(0)
    .getString("base64");

return base64ImageData;
}
```

- Para obter detalhes da API, consulte [InvokeModel](#) a Referência AWS SDK for Java 2.x da API.

Geração de texto com o Jurassic-2 da AI21 Labs

O exemplo de código a seguir mostra como invocar o modelo Jurassic-2 da AI21 Labs no Amazon Bedrock para geração de texto.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque o modelo de base do Jurassic-2 da AI21 Labs de forma assíncrona para gerar texto.

```
/**
 * Asynchronously invokes the AI21 Labs Jurassic-2 model to run an inference
 * based on the provided input.
 *
 * @param prompt The prompt that you want Jurassic to complete.
 * @return The inference response generated by the model.
 */
public static String invokeJurassic2(String prompt) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and default values for Anthropic Claude, refer
     to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     claude.html
     */

    String jurassic2ModelId = "ai21.j2-mid-v1";

    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("temperature", 0.5)
        .put("maxTokens", 200)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(jurassic2ModelId)
```

```

        .contentType("application/json")
        .accept("application/json")
        .build();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                System.out.println("Model invocation failed: " + exception);
            }
        });

    String generatedText = "";
    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        generatedText = responseBody
            .getJSONArray("completions")
            .getJSONObject(0)
            .getJSONObject("data")
            .getString("text");

    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return generatedText;
}

```

Invoque o modelo de base Jurassic-2 da AI21 Labs para gerar texto.

```

/**
 * Invokes the AI21 Labs Jurassic-2 model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Jurassic to complete.
 * @return The generated response.
 */

```

```
public static String invokeJurassic2(String prompt) {
    /*
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for AI21 Labs
Jurassic-2, refer
     * to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-jurassic2.html
     */

    String jurassic2ModelId = "ai21.j2-mid-v1";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

.credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("temperature", 0.5)
        .put("maxTokens", 200)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(jurassic2ModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();

    InvokeModelResponse response = client.invokeModel(request);

    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());

    String generatedText = responseBody
        .getJSONArray("completions")
        .getJSONObject(0)
        .getJSONObject("data")
        .getString("text");

    return generatedText;
}
```

```
}

```

- Para obter detalhes da API, consulte [InvokeModel](#) na Referência AWS SDK for Java 2.x da API.

Geração de texto com o Claude 2 da Anthropic

O exemplo de código a seguir mostra como invocar o modelo Claude 2 da Anthropic no Amazon Bedrock para geração de texto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque o modelo de base Claude 2 da Anthropic de forma assíncrona para gerar texto.

```
/**
 * Asynchronously invokes the Anthropic Claude 2 model to run an inference based
 * on the provided input.
 *
 * @param prompt The prompt that you want Claude to complete.
 * @return The inference response from the model.
 */
public static String invokeClaude(String prompt) {
    /**
     * The different model providers have individual request and response
    formats.
     * For the format, ranges, and default values for Anthropic Claude, refer
    to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    claude.html
     */

    String claudeModelId = "anthropic.claude-v2";

    // Claude requires you to enclose the prompt as follows:
    String enclosedPrompt = "Human: " + prompt + "\n\nAssistant:";

```

```
BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

String payload = new JSONObject()
    .put("prompt", enclosedPrompt)
    .put("max_tokens_to_sample", 200)
    .put("temperature", 0.5)
    .put("stop_sequences", List.of("\n\nHuman:"))
    .toString();

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(claudeModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String generatedText = "";
try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    generatedText = responseBody.getString("completion");
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}

return generatedText;
}
```


Invoque o modelo de base Claude 2 da Anthropic para gerar texto.

```
/**
 * Invokes the Anthropic Claude 2 model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Claude to complete.
 * @return The generated response.
 */
public static String invokeClaude(String prompt) {
    /**
     * The different model providers have individual request and
     response formats.
     * For the format, ranges, and default values for Anthropic Claude,
     refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-claude.html
     */

    String claudeModelId = "anthropic.claude-v2";

    // Claude requires you to enclose the prompt as follows:
    String enclosedPrompt = "Human: " + prompt + "\n\nAssistant:";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", enclosedPrompt)
        .put("max_tokens_to_sample", 200)
        .put("temperature", 0.5)
        .put("stop_sequences", List.of("\n\nHuman:"))
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(claudeModelId)
        .contentType("application/json")
        .accept("application/json")
        .build();
}
```

```
        InvokeModelResponse response = client.invokeModel(request);

        JSONObject responseBody = new
        JSONObject(response.body().asUtf8String());

        String generatedText = responseBody.getString("completion");

        return generatedText;
    }
}
```

- Para obter detalhes da API, consulte [InvokeModel](#) na Referência AWS SDK for Java 2.x da API.

Geração de texto com Claude 2 da Anthropic com fluxo de resposta

O exemplo de código a seguir mostra como invocar o modelo Claude 2 da Anthropic no Amazon Bedrock para geração de texto com fluxo de resposta.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque o modelo Claude 2 da Anthropic e processe o fluxo de resposta.

```
/**
 * Invokes the Anthropic Claude 2 model and processes the response stream.
 *
 * @param prompt The prompt for Claude to complete.
 * @param silent Suppress console output of the individual response stream
 *               chunks.
 * @return The generated response.
 */
public static String invokeClaude(String prompt, boolean silent) {

    BedrockRuntimeAsyncClient client =
    BedrockRuntimeAsyncClient.builder()
        .region(Region.US_EAST_1)
```

```
.credentialsProvider(ProfileCredentialsProvider.create())
    .build();

var finalCompletion = new AtomicReference<>("");

var payload = new JSONObject()
    .put("prompt", "Human: " + prompt + " Assistant:")
    .put("temperature", 0.8)
    .put("max_tokens_to_sample", 300)
    .toString();

var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(payload))
    .modelId("anthropic.claude-v2")
    .contentType("application/json")
    .accept("application/json")
    .build();

var visitor =
InvokeModelWithResponseStreamResponseHandler.Visitor.builder()
    .onChunk(chunk -> {
        var json = new
JSONObject(chunk.bytes().asUtf8String());
        var completion =
json.getString("completion");
        finalCompletion.set(finalCompletion.get() +
completion);

        if (!silent) {
            System.out.print(completion);
        }
    })
    .build();

var handler = InvokeModelWithResponseStreamResponseHandler.builder()
    .onEventStream(stream -> stream.subscribe(event ->
event.accept(visitor)))
    .onComplete(() -> {
    })
    .onError(e -> System.out.println("\n\nError: " +
e.getMessage()))
    .build();

client.invokeModelWithResponseStream(request, handler).join();
```

```
        return finalCompletion.get();
    }
```

- Para obter detalhes da API, consulte [InvokeModelWithResponseStream](#) Referência AWS SDK for Java 2.x da API.

Geração de texto com o Chat do Llama 2 da Meta

O exemplo de código a seguir mostra como invocar o modelo Chat do Llama 2 da Meta no Amazon Bedrock para geração de texto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque o modelo de base Chat do Llama 2 da Meta de forma assíncrona para gerar texto.

```
/**
 * Asynchronously invokes the Meta Llama 2 Chat model to run an inference based
 * on the provided input.
 *
 * @param prompt The prompt that you want Llama 2 to complete.
 * @return The inference response generated by the model.
 */
public static String invokeLlama2(String prompt) {
    /**
     * The different model providers have individual request and response
     formats.
     * For the format, ranges, and default values for Meta Llama 2 Chat, refer
     to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
     meta.
     * html
     */

    String llama2ModelId = "meta.llama2-13b-chat-v1";
```

```
BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

String payload = new JSONObject()
    .put("prompt", prompt)
    .put("max_gen_len", 512)
    .put("temperature", 0.5)
    .put("top_p", 0.9)
    .toString();

InvokeModelRequest request = InvokeModelRequest.builder()
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(llama2ModelId)
    .contentType("application/json")
    .accept("application/json")
    .build();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

String generatedText = "";
try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    generatedText = responseBody.getString("generation");

} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}

return generatedText;
}
```

Invoque o modelo de base do Chat do Llama 2 da Meta para gerar texto.

```
/**
 * Invokes the Meta Llama 2 Chat model to run an inference based on the
provided
 * input.
 *
 * @param prompt The prompt for Llama 2 to complete.
 * @return The generated response.
 */
public static String invokeLlama2(String prompt) {
    /**
     * The different model providers have individual request and
response formats.
     * For the format, ranges, and default values for Meta Llama 2 Chat,
refer to:
     * https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-meta.html
     */

    String llama2ModelId = "meta.llama2-13b-chat-v1";

    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_EAST_1)

        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    String payload = new JSONObject()
        .put("prompt", prompt)
        .put("max_gen_len", 512)
        .put("temperature", 0.5)
        .put("top_p", 0.9)
        .toString();

    InvokeModelRequest request = InvokeModelRequest.builder()
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(llama2ModelId)
        .contentType("application/json")
        .accept("application/json")
```

```
        .build();

        InvokeModelResponse response = client.invokeModel(request);

        JSONObject responseBody = new
        JSONObject(response.body().asUtf8String());

        String generatedText = responseBody.getString("generation");

        return generatedText;
    }
}
```

- Para obter detalhes da API, consulte [InvokeModel](#) na Referência AWS SDK for Java 2.x da API.

Geração de texto com Mistral 7B

O exemplo de código a seguir mostra como invocar o modelo Mistral 7B no Amazon Bedrock para geração de texto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque de forma assíncrona o modelo básico Mistral 7B para gerar texto.

```
/**
 * Asynchronously invokes the Mistral 7B model to run an inference based on the
 * provided input.
 *
 * @param prompt The prompt for Mistral to complete.
 * @return The generated response.
 */
public static List<String> invokeMistral7B(String prompt) {
    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();
```

```
// Mistral instruct models provide optimal results when
// embedding the prompt into the following template:
String instruction = "<s>[INST] " + prompt + " [/INST]";

String modelId = "mistral.mistral-7b-instruct-v0:2";

String payload = new JSONObject()
    .put("prompt", instruction)
    .put("max_tokens", 200)
    .put("temperature", 0.5)
    .toString();

CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request -> request
    .accept("application/json")
    .contentType("application/json")
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(modelId))
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

try {
    InvokeModelResponse response = completableFuture.get();
    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    JSONArray outputs = responseBody.getJSONArray("outputs");

    return IntStream.range(0, outputs.length())
        .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
        .toList();
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    System.err.println(e.getMessage());
} catch (ExecutionException e) {
    System.err.println(e.getMessage());
}

return List.of();
}
```


Invoque o modelo básico Mistral 7B para gerar texto.

```
/**
 * Invokes the Mistral 7B model to run an inference based on the provided
input.
 *
 * @param prompt The prompt for Mistral to complete.
 * @return The generated responses.
 */
public static List<String> invokeMistral7B(String prompt) {
    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Mistral instruct models provide optimal results when
    // embedding the prompt into the following template:
    String instruction = "<s>[INST] " + prompt + " [/INST]";

    String modelId = "mistral.mistral-7b-instruct-v0:2";

    String payload = new JSONObject()
        .put("prompt", instruction)
        .put("max_tokens", 200)
        .put("temperature", 0.5)
        .toString();

    InvokeModelResponse response = client.invokeModel(request -> request
        .accept("application/json")
        .contentType("application/json")
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(modelId));

    JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
    JSONArray outputs = responseBody.getJSONArray("outputs");

    return IntStream.range(0, outputs.length())
        .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
        .toList();
}
```

```
}

```

- Para obter detalhes da API, consulte [InvokeModel](#) na Referência AWS SDK for Java 2.x da API.

Geração de texto com Mixtral 8x7B

O exemplo de código a seguir mostra como invocar o modelo Mixtral 8x7B no Amazon Bedrock para geração de texto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Invoque de forma assíncrona o modelo básico Mistral 8x7B para gerar texto.

```
/**
 * Asynchronously invokes the Mixtral 8x7B model to run an inference based on
 the provided input.
 *
 * @param prompt The prompt for Mixtral to complete.
 * @return The generated response.
 */
public static List<String> invokeMixtral8x7B(String prompt) {
    BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Mistral instruct models provide optimal results when
    // embedding the prompt into the following template:
    String instruction = "<s>[INST] " + prompt + " [/INST]";

    String modelId = "mistral.mixtral-8x7b-instruct-v0:1";

    String payload = new JSONObject()
        .put("prompt", instruction)
        .put("max_tokens", 200)

```

```

        .put("temperature", 0.5)
        .toString();

    CompletableFuture<InvokeModelResponse> completableFuture =
client.invokeModel(request -> request
        .accept("application/json")
        .contentType("application/json")
        .body(SdkBytes.fromUtf8String(payload))
        .modelId(modelId))
    .whenComplete((response, exception) -> {
        if (exception != null) {
            System.out.println("Model invocation failed: " + exception);
        }
    });

    try {
        InvokeModelResponse response = completableFuture.get();
        JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
        JSONArray outputs = responseBody.getJSONArray("outputs");

        return IntStream.range(0, outputs.length())
            .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
            .toList();
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.err.println(e.getMessage());
    } catch (ExecutionException e) {
        System.err.println(e.getMessage());
    }

    return List.of();
}

```

Invoque o modelo básico Mixtral 8x7B para gerar texto.

```

public static List<String> invokeMixtral8x7B(String prompt) {
    BedrockRuntimeClient client = BedrockRuntimeClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();
}

```

```

// Mistral instruct models provide optimal results when
// embedding the prompt into the following template:
String instruction = "<s>[INST] " + prompt + " [/INST]";

String modelId = "mistral.mixtral-8x7b-instruct-v0:1";

String payload = new JSONObject()
    .put("prompt", instruction)
    .put("max_tokens", 200)
    .put("temperature", 0.5)
    .toString();

InvokeModelResponse response = client.invokeModel(request -> request
    .accept("application/json")
    .contentType("application/json")
    .body(SdkBytes.fromUtf8String(payload))
    .modelId(modelId));

JSONObject responseBody = new
JSONObject(response.body().asUtf8String());
JSONArray outputs = responseBody.getJSONArray("outputs");

return IntStream.range(0, outputs.length())
    .mapToObj(i -> outputs.getJSONObject(i).getString("text"))
    .toList();
}

```

- Para obter detalhes da API, consulte [InvokeModel](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação de playground para interagir com os modelos de base do Amazon Bedrock

O exemplo de código a seguir mostra como criar playgrounds para interagir com modelos de base do Amazon Bedrock em diferentes modalidades.

SDK para Java 2.x

O modelo de base (FM) Java Playground é um aplicativo de amostra do Spring Boot que mostra como usar o Amazon Bedrock com Java. Este exemplo mostra como os desenvolvedores Java podem usar o Amazon Bedrock para criar aplicativos habilitados para IA generativa. Você pode

testar e interagir com os modelos de base do Amazon Bedrock usando os três playgrounds a seguir:

- Um playground de texto.
- Um playground de conversa.
- Um playground de imagens.

O exemplo também lista e exibe os modelos de base aos quais tem-se acesso, junto com suas características. Para obter o código-fonte e as instruções de implantação, consulte o projeto em [GitHub](#).

Serviços utilizados neste exemplo

- Amazon Bedrock Runtime

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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

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(b3

-> b5

.cookies(cp -> cp

        .forward(ItemSelection.NONE))

.queryString(true))

                .trustedKeyGroups(b3

-> b3

```

```

    .quantity(1)

    .items(keyGroupId)

    .enabled(true))

> 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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * 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 CreateFunction {

    public static void main(String[] args) {
```

```

    final String usage = ""

        Usage:
            <functionName> <filePath>

        Where:
            functionName - The name of the function to create.\s
            filePath - The path to a file that contains the application
logic for the function.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
        .region(Region.AWS_GLOBAL)
        .build();

    String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
    System.out.println("The function ARN is " + funArn);
    cloudFrontClient.close();
}

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](#) na Referência AWS SDK for Java 2.x da API.

Criar um grupo de chaves

O exemplo de código a seguir mostra como criar um grupo de chaves que você pode usar com URLs e cookies assinados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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 do

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 [AWS Code Examples Repository](#).

O exemplo de código a seguir atualiza uma distribuição para desabilitada, usa um agente de espera 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 da 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 da 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * 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 ModifyDistribution {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <id>\s

                Where:
                id - the id value of the distribution.\s
                """;

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

        String id = args[0];
```

```
CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
    .region(Region.AWS_GLOBAL)
    .build();

modDistribution(cloudFrontClient, id);
cloudFrontClient.close();
}

public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
    try {
        // Get the Distribution to modify.
        GetDistributionRequest disRequest = GetDistributionRequest.builder()
            .id(idVal)
            .build();

        GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
        Distribution disObject = response.distribution();
        DistributionConfig config = disObject.distributionConfig();

        // Create a new DistributionConfig object and add new values to comment
and
        // aliases
        DistributionConfig config1 = DistributionConfig.builder()
            .aliases(config.aliases()) // You can pass in new values here
            .comment("New Comment")
            .cacheBehaviors(config.cacheBehaviors())
            .priceClass(config.priceClass())
            .defaultCacheBehavior(config.defaultCacheBehavior())
            .enabled(config.enabled())
            .callerReference(config.callerReference())
            .logging(config.logging())
            .originGroups(config.originGroups())
            .origins(config.origins())
            .restrictions(config.restrictions())
            .defaultRootObject(config.defaultRootObject())
            .webACLId(config.webACLId())
            .httpVersion(config.httpVersion())
            .viewerCertificate(config.viewerCertificate())
            .customErrorResponses(config.customErrorResponses())
            .build();
```

```
UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
    .distributionConfig(config1)
    .id(disObject.id())
    .ifMatch(response.eTag())
    .build();

cloudFrontClient.updateDistribution(updateDistributionRequest);

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

- 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 o 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 [AWS Code Examples Repository](#).

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

Assinar 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 [AWS Code Examples Repository](#).

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. O `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](#) a 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
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.paginators.ListMetricsIterable;

/**
 * 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 = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
                EC2).\s

                """;

        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)
        .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 [AWS Code Examples Repository](#).

```
public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

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

- Para obter detalhes da API, consulte [PutDashboard](#) da Referência AWS SDK for Java 2.x da API.

Criar 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 [AWS Code Examples Repository](#).

```
public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

        .comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
```

```
        .evaluationPeriods(1)
        .period(10)
        .statistic("Maximum")
        .datapointsToAlarm(1)
        .treatMissingData("ignore")
        .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

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

- 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 [AWS Code Examples Repository](#).

```
public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
String customMetricName =
rootNode.findValue("customMetricName").asText();

SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .stat("Maximum")
    .build();

PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
    .build();

cw.putAnomalyDetector(anomalyDetectorRequest);
System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

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

- Para obter detalhes da API, consulte [PutAnomalyDetector](#) na Referência AWS SDK for Java 2.x da API.

Excluir alarmes

O exemplo de código a seguir mostra como excluir 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DeleteAlarm {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <alarmName>

            Where:
            alarmName - An alarm name to delete (for example, MyAlarm).
            """;

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

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
            .region(region)
            .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                .alarmNames(alarmName)
                .build();
        }
    }
}
```



```
        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

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

- Para obter detalhes da API, consulte [DeleteAlarms](#) na 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 [AWS Code Examples Repository](#).

```
public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
```

```
        .namespace(customMetricNamespace)
        .stat("Maximum")
        .build();

    DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

    cw.deleteAnomalyDetector(request);
    System.out.println("Successfully deleted the Anomaly Detector.");

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

- 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 [AWS Code Examples Repository](#).

```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
```

```
        cw.deleteDashboards(dashboardsRequest);
        System.out.println("Dashboard " + dashboardName + " was successfully deleted.");
    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [DeleteDashboards](#) na 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 alarmes da Amazon.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
```

```
        .alarmName(alarmName)
        .historyItemType(HistoryItemType.ACTION)
        .build();

    DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
    List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
    if (historyItems.isEmpty()) {
        System.out.println("No alarm history data found for " + alarmName +
".".");
    } else {
        for (AlarmHistoryItem item : historyItems) {
            System.out.println("History summary: " + item.historySummary());
            System.out.println("Time stamp: " + item.timestamp());
        }
    }

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

- Para obter detalhes da API, consulte [DescribeAlarmHistory](#) a 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 [AWS Code Examples Repository](#).

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
```

```
typeList.add(AlarmType.METRIC_ALARM);

DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
    .alarmTypes(typeList)
    .maxRecords(10)
    .build();

DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
List<MetricAlarm> alarmList = response.metricAlarms();
for (MetricAlarm alarm : alarmList) {
    System.out.println("Alarm name: " + alarm.alarmName());
    System.out.println("Alarm description: " +
alarm.alarmDescription());
}
} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Para obter detalhes da API, consulte [DescribeAlarms](#) na 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 [AWS Code Examples Repository](#).

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
```

```
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");

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

- Para obter detalhes da API, consulte [DescribeAlarmsForMetric](#) a 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 [AWS Code Examples Repository](#).

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
    }
}
```

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

- Para obter detalhes da API, consulte [DescribeAnomalyDetectors](#) na 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>

                Where:
```



```

        alarmName - An alarm name to disable (for example, MyAlarm).
        """;

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

    String alarmName = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    disableActions(cw, alarmName);
    cw.close();
}

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](#) na 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <alarmName>

                Where:
                alarmName - An alarm name to enable (for example, MyAlarm).
                """;

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

        String alarm = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();
```

```
        enableActions(cw, alarm);
        cw.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
```

```

String myJSON = "{\n" +
    "  \"title\": \"Example Metric Graph\",\n" +
    "  \"view\": \"timeSeries\",\n" +
    "  \"stacked\": false,\n" +
    "  \"period\": 10,\n" +
    "  \"width\": 1400,\n" +
    "  \"height\": 600,\n" +
    "  \"metrics\": [\n" +
    "    [\n" +
    "      \"AWS/Billing\",\n" +
    "      \"EstimatedCharges\",\n" +
    "      \"Currency\",\n" +
    "      \"USD\"\n" +
    "    ]\n" +
    "  ]\n" +
    "}";

GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
    .metricWidget(myJSON)
    .build();

GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
SdkBytes sdkBytes = response.metricWidgetImage();
byte[] bytes = sdkBytes.asByteArray();
File outputFile = new File(fileName);
try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
    outputStream.write(bytes);
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

```

- Para obter detalhes da API, consulte [GetMetricWidgetImage](#) a 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 [AWS Code Examples Repository](#).

```
public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
```

```
        .metricStat(metStat)
        .id("foo2")
        .returnData(true)
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

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

- Para obter detalhes da API, consulte [GetMetricData](#) na 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 [AWS Code Examples Repository](#).

```
public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
    String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

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

```
}  
}
```

- Para obter detalhes da API, consulte [GetMetricStatistics](#) na 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 [AWS Code Examples Repository](#).

```
public static void listDashboards(CloudWatchClient cw) {  
    try {  
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();  
        listRes.stream()  
            .flatMap(r -> r.dashboardEntries().stream())  
            .forEach(entry -> {  
                System.out.println("Dashboard name is: " +  
entry.dashboardName());  
                System.out.println("Dashboard ARN is: " +  
entry.dashboardArn());  
            });  
  
        } catch (CloudWatchException e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Para obter detalhes da API, consulte [ListDashboards](#) na 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 ListMetrics {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
EC2).\s

                """;

        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)
        .build();

    listMets(cw, namespace);
    cw.close();
}

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 [AWS Code Examples Repository](#).

```

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
    }
}

```

```
Instant instant = Instant.parse(time);

MetricDatum datum = MetricDatum.builder()
    .metricName(customMetricName)
    .unit(StandardUnit.NONE)
    .value(1001.00)
    .timestamp(instant)
    .build();

MetricDatum datum2 = MetricDatum.builder()
    .metricName(customMetricName)
    .unit(StandardUnit.NONE)
    .value(1002.00)
    .timestamp(instant)
    .build();

List<MetricDatum> metricDataList = new ArrayList<>();
metricDataList.add(datum);
metricDataList.add(datum2);

PutMetricDataRequest request = PutMetricDataRequest.builder()
    .namespace(customMetricNamespace)
    .metricData(metricDataList)
    .build();

cw.putMetricData(request);
System.out.println("Added metric values for for metric " +
customMetricName);

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

- 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](#).

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
```

```
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
```

```
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * 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 = ""

        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

        Where:
            myDate - The start date to use to get metric statistics. (For
example, 2023-01-11T18:35:24.00Z.)\s
            costDateWeek - The start date to use to get AWS/Billinget
statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
            dashboardName - The name of the dashboard to create.\s
            dashboardJson - The location of a JSON file to use to create a
dashboard. (See Readme file.)\s
            dashboardAdd - The location of a JSON file to use to update a
dashboard. (See Readme file.)\s
            settings - The location of a JSON file from which various values
are read. (See Readme file.)\s
            metricImage - The location of a BMP file that is used to create a
graph.\s

        """;

    if (args.length != 7) {
        System.out.println(usage);
        System.exit(1);
    }

    Region region = Region.US_EAST_1;
    String myDate = args[0];
    String costDateWeek = args[1];
    String dashboardName = args[2];
    String dashboardJson = args[3];
    String dashboardAdd = args[4];
    String settings = args[5];
    String metricImage = args[6];

    Double dataPoint = Double.parseDouble("10.0");
    Scanner sc = new Scanner(System.in);
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();
```



```
System.out.println(DASHES);
System.out.println("Welcome to the Amazon CloudWatch example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
ArrayList<String> list = listNameSpaces(cw);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + list.get(z));
}

String selectedNamespace = "";
String selectedMetrics = "";
int num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedNamespace = list.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedNamespace);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List available metrics within the selected namespace
and select one from the list.");
ArrayList<String> metList = listMets(cw, selectedNamespace);
for (int z = 0; z < 5; z++) {
    int index = z + 1;
    System.out.println("    " + index + ". " + metList.get(z));
}
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    selectedMetrics = metList.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + selectedMetrics);
Dimension myDimension = getSpecificMet(cw, selectedNamespace);
```

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

System.out.println(DASHES);
System.out.println("3. Get statistics for the selected metric over the last
day.");
String metricOption = "";
ArrayList<String> statTypes = new ArrayList<>();
statTypes.add("SampleCount");
statTypes.add("Average");
statTypes.add("Sum");
statTypes.add("Minimum");
statTypes.add("Maximum");

for (int t = 0; t < 5; t++) {
    System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
}
System.out.println("Select a metric statistic by entering a number from the
preceding list:");
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    metricOption = statTypes.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + metricOption);
getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get CloudWatch estimated billing for the last
week.");
getMetricStatistics(cw, costDateWeek);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new CloudWatch dashboard with metrics.");
createDashboardWithMetrics(cw, dashboardName, dashboardJson);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List dashboards using a paginator.");
listDashboards(cw);
```

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

System.out.println(DASHES);
System.out.println("7. Create a new custom metric by adding data to it.");
createNewCustomMetric(cw, dataPoint);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Add an additional metric to the dashboard.");
addMetricToDashboard(cw, dashboardAdd, dashboardName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Create an alarm for the custom metric.");
String alarmName = createAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Describe ten current alarms.");
describeAlarms(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Get current data for new custom metric.");
getCustomMetricData(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Push data into the custom metric to trigger the
alarm.");
addMetricDataForAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
checkForMetricAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Get alarm history for the new alarm.");
getAlarmHistory(cw, settings, myDate);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("15. Add an anomaly detector for the custom metric.");
addAnomalyDetector(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Describe current anomaly detectors.");
describeAnomalyDetectors(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Get a metric image for the custom metric.");
getAndOpenMetricImage(cw, metricImage);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("18. Clean up the Amazon CloudWatch resources.");
deleteDashboard(cw, dashboardName);
deleteCWAlarm(cw, alarmName);
deleteAnomalyDetector(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Amazon CloudWatch example scenario is complete.");
System.out.println(DASHES);
cw.close();
}

public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
```

```
        .build());

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.println("Successfully deleted alarm " + alarmName);

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

public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
    }
}
```

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

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +
            "  \"period\": 10,\n" +
            "  \"width\": 1400,\n" +
            "  \"height\": 600,\n" +
            "  \"metrics\": [\n" +
            "    [\n" +
            "      \"AWS/Billing\",\n" +
            "      \"EstimatedCharges\",\n" +
            "      \"Currency\",\n" +
            "      \"USD\"\n" +
            "    ]\n" +
            "  ]\n" +
            "}";

        GetMetricWidgetImageRequest imageRequest =
            GetMetricWidgetImageRequest.builder()
                .metricWidget(myJSON)
                .build();

        GetMetricWidgetImageResponse response =
            cw.getMetricWidgetImage(imageRequest);
        SdkBytes sdkBytes = response.metricWidgetImage();
        byte[] bytes = sdkBytes.asByteArray();
        File outputFile = new File(fileName);
        try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
            outputStream.write(bytes);
        }

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

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
            .maxResults(10)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
        List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
        for (AnomalyDetector detector : anomalyDetectorList) {
            System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
            System.out.println("State: " + detector.stateValue());
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .stat("Maximum")
    .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
    .build();

        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

    public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();

            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                .startDate(start)
                .endDate(endDate)
                .alarmName(alarmName)
                .historyItemType(HistoryItemType.ACTION)
                .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();

```



```

        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
                ".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)

```

```

        System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
    } else
        System.out.println("Alarm state found for " + customMetricName +
".");

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

public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
    }
}

```

```
metricDataList.add(datum2);

PutMetricDataRequest request = PutMetricDataRequest.builder()
    .namespace(customMetricNamespace)
    .metricData(metricDataList)
    .build();

cw.putMetricData(request);
System.out.println("Added metric values for for metric " +
customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
```

```
        .metric(met)
        .build();

MetricDataQuery dataQuery = MetricDataQuery.builder()
    .metricStat(metStat)
    .id("foo2")
    .returnData(true)
    .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();
```

```

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")

```

```
        .datapointsToAlarm(1)
        .treatMissingData("ignore")
        .build();

    cw.putMetricAlarm(alarmRequest);
    System.out.println(alarmName + " was successfully created!");
    return alarmName;

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

public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully updated.");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
{
    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);
```

```
        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric PAGES_VISITED");

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

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

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

    public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
        try {
```

```
PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
    .dashboardName(dashboardName)
    .dashboardBody(readFileAsString(fileName))
    .build();

PutDashboardResponse response = cw.putDashboard(dashboardRequest);
System.out.println(dashboardName + " was successfully created.");
List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
if (messages.isEmpty()) {
    System.out.println("There are no messages in the new Dashboard");
} else {
    for (DashboardValidationMessage message : messages) {
        System.out.println("Message is: " + message.message());
    }
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String readFileAsString(String file) throws IOException {
    return new String(Files.readAllBytes(Paths.get(file)));
}

public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
    try {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
```



```
        .statistics(Statistic.MAXIMUM)
        .startTime(start)
        .endTime(endDate)
        .period(86400)
        .build();

    GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
    List<Datapoint> data = response.datapoints();
    if (!data.isEmpty()) {
        for (Datapoint datapoint : data) {
            System.out
                .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
        }
    } else {
        System.out.println("The returned data list is empty");
    }

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

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400)
            .statistics(Statistic.fromValue(metricOption))
            .build();
```

```

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

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

public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsResponse response = cw.listMetrics(request);
        List<Metric> myList = response.metrics();
        Metric metric = myList.get(0);
        return metric.dimensions().get(0);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
{
    try {
        ArrayList<String> metList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

```

```

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> metList.add(metrics.metricName()));

        return metList;

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
    try {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> {
                String data = metrics.namespace();
                if (!nameSpaceList.contains(data)) {
                    nameSpaceList.add(data);
                }
            });

        return nameSpaceList;
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
}

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
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.Target;

/**
 * To run this Java V2 code example, ensure that you have setup 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 PutTargets {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <ruleName> <functionArn> <targetId>\s

            Where:
                ruleName - A rule name (for example, myrule).
                functionArn - An AWS Lambda function ARN (for example,
                arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                targetId - A target id value.

            """;
```

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

    String ruleName = args[0];
    String functionArn = args[1];
    String targetId = args[2];
    CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
        .build();

    putCWTargets(cwe, ruleName, functionArn, targetId);
    cwe.close();
}

public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
String functionArn, String targetId) {
    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);

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

- Para obter detalhes da API, consulte [PutTargets](#) na 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 do Amazon CloudWatch Events.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 PutRule {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <ruleName> roleArn\s

                Where:
                ruleName - A rule name (for example, myrule).
                roleArn - A role ARN value (for example,
                arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String ruleName = args[0];
    String roleArn = args[1];
    CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
        .build();

    putCWRule(cwe, ruleName, roleArn);
    cwe.close();
}

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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 PutEvents {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <resourceArn>

                Where:
                resourceArn - An Amazon Resource Name (ARN) related to the
events.

                """;

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

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();
```

```
        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    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](#) na 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
```

```

* '111111111111' with your account details.
* For more information, see "Subscription Filters with AWS Lambda" in the
* Amazon CloudWatch Logs Guide.
*
*
* Also, 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 PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <pattern> <logGroup> <functionArn>\s

            Where:
                filter - A filter name (for example, myfilter).
                pattern - A filter pattern (for example, ERROR).
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
            .region(region)
            .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
    }
}

```

```
        cwl.close();
    }

    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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
  software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * 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 DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <filter> <logGroup>

            Where:
            filter - The name of the subscription filter (for example,
MyFilter).
            logGroup - The name of the log group. (for example, testgroup).
            ""

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
```

```
        .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * 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 DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
            <logGroup>

            Where:
            logGroup - A log group name (for example, myloggroup).
            """;

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

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        describeFilters(logs, logGroup);
        logs.close();
    }
}
```



```
public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
    try {
        boolean done = false;
        String newToken = null;

        while (!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .logGroupName(logGroup)
                    .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .nextToken(newToken)
                    .logGroupName(logGroup)
                    .limit(1).build();
                response = logs.describeSubscriptionFilters(request);
            }

            for (SubscriptionFilter filter : response.subscriptionFilters()) {
                System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
                    filter.filterName(),
                    filter.filterPattern(),
                    filter.destinationArn());
            }

            if (response.nextToken() == null) {
                done = true;
            } else {
                newToken = response.nextToken();
            }
        }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Done");
}
```

```
}

```

- Para obter detalhes da API, consulte [DescribeSubscriptionFilters](#) na Referência AWS SDK for Java 2.x da API.

Iniciar uma sessão Live Tail

O exemplo de código a seguir mostra como iniciar uma sessão do Live Tail para um grupo de logs/fluxo de logs existente.

SDK para Java 2.x

Inclua os arquivos necessários.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Gerencie os eventos da sessão do Live Tail.

```
private static StartLiveTailResponseHandler
getStartLiveTailResponseStreamHandler(
    AtomicReference<Subscription> subscriptionAtomicReference) {
    return StartLiveTailResponseHandler.builder()
        .onResponse(r -> System.out.println("Received initial response"))
```

```

        .onError(throwable -> {
            CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        })
        .subscriber(() -> new FlowableSubscriber<>() {
            @Override
            public void onSubscribe(@NonNull Subscription s) {
                subscriptionAtomicReference.set(s);
                s.request(Long.MAX_VALUE);
            }

            @Override
            public void onNext(StartLiveTailResponseStream event) {
                if (event instanceof LiveTailSessionStart) {
                    LiveTailSessionStart sessionStart = (LiveTailSessionStart)
event;

                    System.out.println(sessionStart);
                } else if (event instanceof LiveTailSessionUpdate) {
                    LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                    List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                    logEvents.forEach(e -> {
                        long timestamp = e.timestamp();
                        Date date = new Date(timestamp);
                        System.out.println "[" + date + "]" + e.message());
                    });
                } else {
                    throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                }
            }

            @Override
            public void onError(Throwable throwable) {
                System.out.println(throwable.getMessage());
                System.exit(1);
            }

            @Override
            public void onComplete() {
                System.out.println("Completed Streaming Session");
            }
        });

```

```

        }
    })
    .build();
}

```

Inicie a sessão do Live Tail.

```

CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
    CloudWatchLogsAsyncClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

StartLiveTailRequest request =
    StartLiveTailRequest.builder()
        .logGroupIdentifiers(logGroupIdentifiers)
        .logStreamNames(logStreamNames)
        .logEventFilterPattern(logEventFilterPattern)
        .build();

/* Create a reference to store the subscription */
final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));

```

Interrompa a sessão do Live Tail após um período decorrido.

```

/* Set a timeout for the session and cancel the subscription. This will:
 * 1). Close the stream
 * 2). Stop the Live Tail session
 */
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
if (subscriptionAtomicReference.get() != null) {
    subscriptionAtomicReference.get().cancel();
    System.out.println("Subscription to stream closed");
}

```

- Para obter detalhes da API, consulte [StartLiveTaila](#) Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Cognito Identity usando o SDK para 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.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 banco de identidades do

O exemplo de código a seguir mostra como criar um banco de identidades 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](#).

```
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;

/**
 * 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 CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <identityPoolName>\s

            Where:
                identityPoolName - The name to give your identity pool.
            """;

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

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    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 "";
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API do AWS SDK for Java 2.x .
 - [CreateIdentityPool](#)
 - [ListIdentityPools](#)

Excluir um grupo de identidades

O exemplo de código a seguir mostra como excluir um banco de identidades do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
```

```
* 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 DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <identityPoolId>\s

            Where:
                identityPoolId - The Id value of your identity pool.
            """;

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

        String identityPoolId = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        deleteIdPool(cognitoIdClient, identityPoolId);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
identityPoolId) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
DeleteIdentityPoolRequest.builder()
                .identityPoolId(identityPoolId)
                .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
        }
    }
}
```



```
        System.out.println("Done");

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

- Para obter detalhes da API, consulte [DeletIdentityPool](#) a Referência AWS SDK for Java 2.x da API.

Obter credenciais para uma identidade

O exemplo de código a seguir mostra como obter credenciais para uma identidade do Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
REGION:GUID.
            """;

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

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    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);
        }
    }
}
```

- Para obter detalhes da API, consulte [GetCredentialsForIdentity](#) a Referência AWS SDK for Java 2.x da API.

Listar grupos de identidades

O exemplo de código a seguir mostra como obter uma lista do banco de identidades 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](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
    software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * 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 ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
                ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
                cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API do AWS SDK for Java 2.x .
 - [CreateIdentityPool](#)
 - [ListIdentityPools](#)

Exemplos do provedor de identidade do Amazon Cognito usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 Cognito

Os exemplos de código a seguir mostram como começar a usar o Amazon Cognito.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;
```

```
/**
 * 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 ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    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);
        }
    }
}
```

- Para obter detalhes da API, consulte [ListUserPools](#) na Referência AWS SDK for Java 2.x da API.

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 [AWS Code Examples Repository](#).

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

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

- Para obter detalhes da API, consulte [ConfirmSignUp](#) na Referência AWS SDK for Java 2.x da API.

Criar um grupo de usuários

O exemplo de código a seguir mostra como criar um grupo de 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreateUserPool {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolName>\s

            Where:
                userPoolName - The name to give your user pool when it's
created.

        "";
```



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

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                .poolName(userPoolName)
                .build();

            CreateUserPoolResponse response = cognitoClient.createUserPool(request);
            return response.userPool().id();

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

- Para obter detalhes da API, consulte [CreateUserPool](#) na Referência AWS SDK for Java 2.x da API.

Criar um cliente da aplicação

O exemplo de código a seguir mostra como criar um cliente da aplicação do grupo de 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 [AWS Code Examples Repository](#).

```
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;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * 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 CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clientName> <userPoolId>\s

            Where:
                clientName - The name for the user pool client to create.
```

```
        userPoolId - The ID for the user pool.
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String clientName = args[0];
    String userPoolId = args[1];
    CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPoolClient(cognitoClient, clientName, userPoolId);
    cognitoClient.close();
}

public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
String clientName,
    String userPoolId) {
    try {
        CreateUserPoolClientRequest request =
CreateUserPoolClientRequest.builder()
            .clientName(clientName)
            .userPoolId(userPoolId)
            .build();

        CreateUserPoolClientResponse response =
cognitoClient.createUserPoolClient(request);
        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);
    }
}
}
```

- Para obter detalhes da API, consulte [CreateUserPoolClient](#) na 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 uma aplicação 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 [AWS Code Examples Repository](#).

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

- Para obter detalhes da API, consulte [AssociateSoftwareToken](#) na 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 [AWS Code Examples Repository](#).

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
    String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

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

- Para obter detalhes da API, consulte [AdminGetUser](#) Referência AWS SDK for Java 2.x da API.

Listar os grupos de usuários

O exemplo de código a seguir mostra como listar os grupos de 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
        CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    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);
        }
    }
}

```

- Para obter detalhes da API, consulte [ListUserPools](#) a 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 [AWS Code Examples Repository](#).

```

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.ListUsersRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * 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 ListUsers {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <userPoolId>\s

            Where:
                userPoolId - The ID given to your user pool when it's created.
                """;

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

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

- 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 [AWS Code Examples Repository](#).

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
    String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}
```

- Para obter detalhes da API, consulte [AdminRespondToAuthChallenge](#) Referência AWS SDK for Java 2.x da API.

Inscrever um usuário

O exemplo de código a seguir mostra como inscrever 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 [AWS Code Examples Repository](#).

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);
    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

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

- 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 credenciais de administrador.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

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

    return null;
}
```

- 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 uma aplicação 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 [AWS Code Examples Repository](#).

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

- Para obter detalhes da API, consulte [VerifySoftwareToken](#) Referência AWS SDK for Java 2.x da API.

Cenários

Inscriver 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](#).

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeResponse;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
```

```
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResponse;
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * 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 = ""

            Usage:
                <clientId> <poolId>

            Where:
                clientId - The app client Id value that you can get from the AWS
CDK script.
                poolId - The pool Id that you can get from the AWS CDK script.\s
            """;

        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)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

```

```
System.out.println("*** Enter your password");
String password = in.nextLine();

System.out.println("*** Enter your email");
String email = in.nextLine();

System.out.println("1. Signing up " + userName);
signUp(identityProviderClient, clientId, userName, password, email);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Getting " + userName + " in the user pool");
getAdminUser(identityProviderClient, userName, poolId);

System.out
    .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
System.out.println(DASHES);

System.out.println(DASHES);
String ans = in.nextLine();

if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
    poolId);
String mySession = authResponse.session();
```

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

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
```

```
        String userName, String clientId, String mfaCode, String session) {
    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
        .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
        + respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

```
public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
             String clientId, String userName, String password, String userPoolId) {
    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                        .clientId(clientId)
                        .userPoolId(userPoolId)
                        .authParameters(authParameters)
                        .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                        .build();

        AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName());
        return response;

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

    return null;
}

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                        .session(session)
                        .build();

    AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
        .associateSoftwareToken(softwareTokenRequest);
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
    String userName) {
    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

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

public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
    String userName) {
    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
    String password, String email) {
    AttributeType userAttrs = AttributeType.builder()
```

```
        .name("email")
        .value(email)
        .build();

List<AttributeType> userAttrsList = new ArrayList<>();
userAttrsList.add(userAttrs);
try {
    SignUpRequest signUpRequest = SignUpRequest.builder()
        .userAttributes(userAttrsList)
        .username(userName)
        .clientId(clientId)
        .password(password)
        .build();

    identityProviderClient.signUp(signUpRequest);
    System.out.println("User has been signed up ");

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

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
String poolId) {
    try {
        AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status " + response.userStatusAsString());

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API do AWS SDK for Java 2.x .
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

Exemplos do Amazon Comprehend usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 do Amazon Comprehend.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
    software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
    software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, 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 DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <dataAccessRoleArn> <s3Uri> <documentClassifierName>
```

```
        Where:
        dataAccessRoleArn - The ARN value of the role used for this
operation.
        s3Uri - The Amazon S3 bucket that contains the CSV file.
        documentClassifierName - The name of the document classifier.
        """";

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

    String dataAccessRoleArn = args[0];
    String s3Uri = args[1];
    String documentClassifierName = args[2];

    Region region = Region.US_EAST_1;
    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
    comClient.close();
}

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 [AWS Code Examples Repository](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * 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 DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    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](#) a Referência AWS SDK for Java 2.x da API.

Detectar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * 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 DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
```

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

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    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.

Detectar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * 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 DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
        blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
        Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
            .region(region)
            .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    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 [DetectSyntax](#) Referência AWS SDK for Java 2.x da API.

Detectar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
```



```
import
  software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * 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 DetectLanguage {
  public static void main(String[] args) {
    // Specify French text - "It is raining today in Seattle".
    String text = "Il pleut aujourd'hui à Seattle";
    Region region = Region.US_EAST_1;

    ComprehendClient comClient = ComprehendClient.builder()
      .region(region)
      .build();

    System.out.println("Calling DetectDominantLanguage");
    detectTheDominantLanguage(comClient, text);
    comClient.close();
  }

  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.

Detectar o sentimento de um documento

O exemplo de código a seguir mostra como detectar o sentimento 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * 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 DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
        July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
```

```
blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
Seattle - based companies are Starbucks and Boeing.";
    Region region = Region.US_EAST_1;
    ComprehendClient comClient = ComprehendClient.builder()
        .region(region)
        .build();

    System.out.println("Calling DetectSentiment");
    detectSentiments(comClient, text);
    comClient.close();
}

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 para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

/**
 * 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 ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
    }
}
```

```
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
listAllTables(ddb);
ddb.close();
}

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!");
    }
}
```

- Para obter detalhes da API, consulte [ListTables](#) Referência AWS SDK for Java 2.x da API.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
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.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
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.waiters.DynamoDbWaiter;
```

```
/**
 * 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 CreateTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example,
Artist).

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    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(10L)
                .writeCapacityUnits(10L)
                .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 [CreateTable](#) 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DeleteTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table to delete (for example,
                Music3).

                **Warning** This program will delete the table that you specify!
                """;

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

```
    }

    String tableName = args[0];
    System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    deleteDynamoDBTable(ddb, tableName);
    ddb.close();
}

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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DeleteItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyval>

                Where:
                tableName - The Amazon DynamoDB table to delete the item from
(for example, Music3).
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                keyval - The key value that represents the item to delete (for
example, Famous Band).
                """;

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

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
```

```
        .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    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](#) Referência AWS SDK for Java 2.x da API.

Obter um lote de itens

O exemplo de código a seguir mostra como obter 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 [AWS Code Examples Repository](#).

mostra como obter itens em lote usando o cliente de serviço.

```
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.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * 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 BatchReadItems {
    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
                .region(region)
                .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
```

```

key1.put("Artist", AttributeValue.builder().s("Artist1").build());

Map<String, AttributeValue> key2 = new HashMap<>();
key2.put("Artist", AttributeValue.builder().s("Artist2").build());

// Construct the batchGetItem request.
Map<String, KeysAndAttributes> requestItems = new HashMap<>();
requestItems.put(tableName, KeysAndAttributes.builder()
    .keys(List.of(key1, key2))
    .projectionExpression("Artist, SongTitle")
    .build());

BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
    .requestItems(requestItems)
    .build();

// Make the batchGetItem request.
BatchGetItemResponse batchGetItemResponse =
dynamoDbClient.batchGetItem(batchGetItemRequest);

// Extract and print the retrieved items.
Map<String, List<Map<String, AttributeValue>>> responses =
batchGetItemResponse.responses();
if (responses.containsKey(tableName)) {
    List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
    for (Map<String, AttributeValue> item : musicItems) {
        System.out.println("Artist: " + item.get("Artist").s() +
            ", SongTitle: " + item.get("SongTitle").s());
    }
} else {
    System.out.println("No items retrieved.");
}
}
}

```

mostra como obter itens em lote usando o cliente de serviço e um paginador.

```

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.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;

```

```
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music).\s
            """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
```

```

        .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
            .flatMap(response -> response.responses().getOrDefault(tableName,
Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}

```

- Para obter detalhes da API, consulte [BatchGetItem](#) 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 [AWS Code Examples Repository](#).

Obtém um item de uma tabela usando DynamoDbClient o.

```

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;

/**
 * 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 get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
* its better practice to use the
* Enhanced Client, see the EnhancedGetItem example.
*/
public class GetItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal>

            Where:
                tableName - The Amazon DynamoDB table from which an item is
retrieved (for example, Music3).\s
                key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                keyval - The key value that represents the item to get (for
example, Famous Band).
            """;

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

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Retrieving item \"%s\" from \"%s\"\\n", keyVal,
tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        getDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }
}
```

```
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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DescribeTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName>

                Where:
                tableName - The Amazon DynamoDB table to get information about
                (for example, Music3).
                """;

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

```
String tableName = args[0];
System.out.format("Getting description for %s\n\n", tableName);
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

describeDynamoDBTable(ddb, tableName);
ddb.close();
}

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());
            System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

            ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits());
            System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits());

            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 [AWS Code Examples Repository](#).

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

/**
 * 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 ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
```

```
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();
listAllTables(ddb);
ddb.close();
}

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!");
}
```

```
}  
}
```

- 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 [AWS Code Examples Repository](#).

Coloca um item em uma tabela usando [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;  
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;  
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;  
import java.util.HashMap;  
  
/**  
 * 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 place items into an Amazon DynamoDB table using the AWS SDK for Java V2,  
 * its better practice to use the  
 * Enhanced Client. See the EnhancedPutItem example.  
 */  
public class PutItem {
```

```

public static void main(String[] args) {
    final String usage = ""

        Usage:
            <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
<awardsval> <Songtitle> <songtitleval>

        Where:
            tableName - The Amazon DynamoDB table in which an item is placed
(for example, Music3).
            key - The key used in the Amazon DynamoDB table (for example,
Artist).
            keyval - The key value that represents the item to get (for
example, Famous Band).
            albumTitle - The Album title (for example, AlbumTitle).
AlbumTitleValue - The name of the album (for example, Songs
About Life ).
            Awards - The awards column (for example, Awards).
AwardVal - The value of the awards (for example, 10).
            SongTitle - The song title (for example, SongTitle).
SongTitleVal - The value of the song title (for example, Happy
Day).

        **Warning** This program will place an item that you specify into a
table!

        """;

    if (args.length != 9) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String key = args[1];
    String keyVal = args[2];
    String albumTitle = args[3];
    String albumTitleValue = args[4];
    String awards = args[5];
    String awardVal = args[6];
    String songTitle = args[7];
    String songTitleVal = args[8];

    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)

```



```

        .build();

        putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
            songTitleVal);
        System.out.println("Done!");
        ddb.close();
    }

    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 [AWS Code Examples Repository](#).

Consulta uma tabela usando [DynamoDbClient](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * 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 query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
```

```
* Enhanced Client. See the EnhancedQueryRecords example.
*/
public class Query {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <partitionKeyName> <partitionKeyVal>

            Where:
                tableName - The Amazon DynamoDB table to put the item in (for
example, Music3).
                partitionKeyName - The partition key name of the Amazon DynamoDB
table (for example, Artist).
                partitionKeyVal - The value of the partition key that should
match (for example, Famous Band).
            """;

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

        String tableName = args[0];
        String partitionKeyName = args[1];
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
        String partitionAlias = "#a";

        System.out.format("Querying %s", tableName);
        System.out.println("");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
partitionAlias);
        System.out.println("There were " + count + " record(s) returned");
        ddb.close();
    }
}
```

```
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.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
```

```
import java.util.Map;

/**
 * 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
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * year-index. For more information, see:
 *
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
 */
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }

    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 verificar 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 [AWS Code Examples Repository](#).

Escaneia uma tabela do Amazon DynamoDB [DynamoDbClient](#) usando.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * 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 scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
* its better practice to use the
* Enhanced Client, See the EnhancedScanRecords example.
*/
```

```
public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to get information from
(for example, Music3).
                """;

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

        String tableName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        scanItems(ddb, tableName);
        ddb.close();
    }

    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 [AWS Code Examples Repository](#).

Atualiza um item em uma tabela usando [DynamoDbClient](#).

```
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.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;
```



```
/**
 * 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 update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key> <keyVal> <name> <updateVal>

            Where:
                tableName - The Amazon DynamoDB table (for example, Music3).
                key - The name of the key in the table (for example, Artist).
                keyVal - The value of the key (for example, Famous Band).
                name - The name of the column where the value is updated (for
example, Awards).
                updateVal - The value used to update an item (for example, 14).
            Example:
                UpdateItem Music3 Artist Famous Band Awards 14
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
```

```
        .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    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 (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 [AWS Code Examples Repository](#).

Insere muitos itens em uma tabela usando o cliente de serviço.

```
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.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * 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 BatchWriteItems {
    public static void main(String[] args){
        final String usage = ""

                Usage:
                <tableName>

                Where:
```

```

        tableName - The Amazon DynamoDB table (for example, Music).\s
        """;

String tableName = "Music";
Region region = Region.US_EAST_1;
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .region(region)
    .build();

addBatchItems(dynamoDbClient, tableName);
}

public static void addBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
    // Specify the updates you want to perform.
    List<WriteRequest> writeRequests = new ArrayList<>();

    // Set item 1.
    Map<String, AttributeValue> item1Attributes = new HashMap<>();
    item1Attributes.put("Artist",
AttributeValue.builder().s("Artist1").build());
    item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
    item1Attributes.put("Comments", AttributeValue.builder().s("Great
song!").build());
    item1Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle1").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attribut

    // Set item 2.
    Map<String, AttributeValue> item2Attributes = new HashMap<>();
    item2Attributes.put("Artist",
AttributeValue.builder().s("Artist2").build());
    item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
    item2Attributes.put("Comments", AttributeValue.builder().s("Nice
melody.").build());
    item2Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle2").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attribut

    try {
        // Create the BatchWriteItemRequest.

```

```

        BatchWriteItemRequest batchWriteItemRequest =
BatchWriteItemRequest.builder()
    .requestItems(Map.of(tableName, writeRequests))
    .build();

        // Execute the BatchWriteItem operation.
        BatchWriteItemResponse batchWriteItemResponse =
dynamoDbClient.batchWriteItem(batchWriteItemRequest);

        // Process the response.
        System.out.println("Batch write successful: " + batchWriteItemResponse);

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

```

Insira vários itens em uma tabela usando o cliente aprimorado.

```

import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 * with these columns:
 * - id - the id of the record that is the key
 * - custName - the customer name

```

```
* - email - the email value
* - registrationDate - an instant value when the item was added to the table
*
* Also, ensure that you have 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 EnhancedBatchWriteItems {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
        putBatchRecords(enhancedClient);
        ddb.close();
    }

    public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
        try {
            DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                TableSchema.fromBean(Customer.class));
            DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                TableSchema.fromBean(Music.class));
            LocalDate localDate = LocalDate.parse("2020-04-07");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            Customer record2 = new Customer();
            record2.setCustName("Fred Pink");
            record2.setId("id110");
            record2.setEmail("fredp@noserver.com");
            record2.setRegistrationDate(instant);

            Customer record3 = new Customer();
            record3.setCustName("Susan Pink");
```

```

        record3.setId("id120");
        record3.setEmail("spink@noserver.com");
        record3.setRegistrationDate(instant);

        Customer record4 = new Customer();
        record4.setCustName("Jerry orange");
        record4.setId("id101");
        record4.setEmail("jorange@noserver.com");
        record4.setRegistrationDate(instant);

        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                .builder()
                                .writeBatches(
WriteBatch.builder(Customer.class) // add items to the Customer

        // table

        .mappedTableResource(customerMappedTable)

        .addPutItem(builder -> builder.item(record2))

        .addPutItem(builder -> builder.item(record3))

        .addPutItem(builder -> builder.item(record4))

                                                                .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

        // table

        .mappedTableResource(musicMappedTable)

        .addDeleteItem(builder -> builder.key(

            Key.builder().partitionValue(

                "Famous Band")

                .build()))

                                                                .build())

                                                                .build();

```

```
        // Add three items to the Customer table and delete one item
from the Music
        // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
        System.out.println("done");

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

- Para obter detalhes da API, consulte [BatchWriteItem](#) na 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.
- Grave 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(10L)
            .writeCapacityUnits(10L)
            .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 = ""

                Usage:
                <fileName>

```

```
        Where:
            fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
        """;

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

    String tableName = "Movies";
    String fileName = args[0];
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .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(10L)
        .writeCapacityUnits(10L)
        .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 da 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 [AWS Code Examples Repository](#).

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQLBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .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](#) a 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 [AWS Code Examples Repository](#).

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
```

```
final String usage = ""

    Usage:
        <fileName>

    Where:
        fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.
        """;

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

String fileName = args[0];
String tableName = "MoviesPartiQ";
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

System.out.println(
    "***** Creating an Amazon DynamoDB table named MoviesPartiQ with a
key named year and a sort key named title.");
createTable(ddb, tableName);

System.out.println("***** Loading data into the MoviesPartiQ table.");
loadData(ddb, fileName);

System.out.println("***** Getting data from the MoviesPartiQ table.");
getItem(ddb);

System.out.println("***** Putting a record into the MoviesPartiQ table.");
putRecord(ddb);

System.out.println("***** Updating a record.");
updateTableItem(ddb);

System.out.println("***** Querying the movies released in 2013.");
queryTable(ddb);

System.out.println("***** Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
```

```
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
```



```
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

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

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

        // Add 200 movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf(year))
```

```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s(title)
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s(info)
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        // Insert the movie into the Amazon DynamoDB table.
        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added Movie " + title);

        parameters.remove(att1);
        parameters.remove(att2);
        parameters.remove(att3);
        t++;
    }
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The Perks of Being a Wallflower")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        ExecuteStatementResponse response = executeStatementRequest(ddb,
            sqlStatement, parameters);
```

```
        System.out.println("ExecuteStatement successful: " +
response.toString());

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

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added new movie.");

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

public static void updateTableItem(DynamoDbClient ddb) {
```

```
String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
List<AttributeValue> parameters = new ArrayList<>();
AttributeValue att1 = AttributeValue.builder()
    .n(String.valueOf("2013"))
    .build();

AttributeValue att2 = AttributeValue.builder()
    .s("The East")
    .build();

parameters.add(att1);
parameters.add(att2);

try {
    executeStatementRequest(ddb, sqlStatement, parameters);

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Item was updated!");
}

// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

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

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

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement,
    List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse
executeStatementResult) {
    System.out.println("ExecuteStatement successful: " +
executeStatementResult.toString());
}
}
```

- Para obter detalhes da API, consulte [ExecuteStatement](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon EC2 usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
```

```
        .flatMap(r -> r.securityGroups().stream())
        .forEach(group -> System.out
            .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [DescribeSecurityGroups](#) na 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 [AWS Code Examples Repository](#).

```
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 "";
}
```

- Para obter detalhes da API, consulte [AllocateAddress](#) a 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 [AWS Code Examples Repository](#).

```
public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();
    }
```



```
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obter detalhes da API, consulte [AssociateAddress](#) Referência AWS SDK for Java 2.x da API.

Criar 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 [AWS Code Examples Repository](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();
```

```
    IpPermission ipPerm = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(80)
        .fromPort(80)
        .ipRanges(ipRange)
        .build();

    IpPermission ipPerm2 = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(22)
        .fromPort(22)
        .ipRanges(ipRange)
        .build();

    AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

    ec2.authorizeSecurityGroupIngress(authRequest);
    System.out.println("Successfully added ingress policy to security group
" + groupName);
    return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obter detalhes da API, consulte [CreateSecurityGroup](#) 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 do Amazon EC2.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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 [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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
 */
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <name> <amiId>

                Where:
                name - An instance name value that you can obtain from the AWS
                Console (for example, ami-xxxxxx5c8b987b1a0).\s
                amiId - An Amazon Machine Image (AMI) value that you can obtain
                from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
    }
}
```

```
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String name = args[0];
    String amiId = args[1];
    Region region = Region.US_EAST_1;
    Ec2Client ec2 = Ec2Client.builder()
        .region(region)
        .build();

    String instanceId = createEC2Instance(ec2, name, amiId);
    System.out.println("The Amazon EC2 Instance ID is " + instanceId);
    ec2.close();
}

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();

    // Use a waiter to wait until the instance is running.
    System.out.println("Going to start an EC2 instance using a waiter");
    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceIdVal = response.instances().get(0).instanceId();
    ec2.waitFor().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceIdVal)
        .tags(tag)
        .build();

    try {
```

```
        ec2.createTags(tagRequest);
        System.out.printf("Successfully started EC2 Instance %s based on AMI
%s", instanceIdVal, amiId);
        return instanceIdVal;

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Para obter detalhes da API, consulte [RunInstances](#) a Referência AWS SDK for Java 2.x da API.

Exclua 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 [AWS Code Examples Repository](#).

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
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 [AWS Code Examples Repository](#).

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + 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 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 [AWS Code Examples Repository](#).

```
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.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
 * 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 DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
```



```

        .flatMap(r -> r.reservations().stream())
        .flatMap(reservation -> reservation.instances().stream())
        .forEach(instance -> {
            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());
        });

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorCode());
        System.exit(1);
    }
}
}
}

```

- Para obter detalhes da API, consulte [DescribeInstances](#) a 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 [AWS Code Examples Repository](#).

```

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()

```

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

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 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 [AWS Code Examples Repository](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
```

```
        .forEach(group -> System.out
            .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } 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 tipos de 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 [AWS Code Examples Repository](#).

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .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());
        System.out.println("Instance type is " +
type.instanceType().toString());
        instanceType = type.instanceType().toString();
        if (instanceType.compareTo("t2.2xlarge") == 0){
            return instanceType;
        }
    }

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

- Para obter detalhes da API, consulte [DescribeInstanceTypes](#) a Referência AWS SDK for Java 2.x da API.

Listar pares de chaves de segurança

O exemplo de código a seguir mostra como listar 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 [AWS Code Examples Repository](#).

```
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);  
    }  
}
```

- Para obter detalhes da API, consulte [DescribeKeyPairs](#) 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 [AWS Code Examples Repository](#).

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {  
    try {  
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()  
            .allocationId(allocId)  
            .build();  
  
        ec2.releaseAddress(request);  
        System.out.println("Successfully released Elastic IP address " +  
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 [AWS Code Examples Repository](#).

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
```

```
        .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- 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 [AWS Code Examples Repository](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
```

```
StartInstancesRequest request = StartInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
ec2.startInstances(request);
DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Successfully started instance " + instanceId);
}
```

- Para obter detalhes da API, consulte [StartInstances](#) na Referência AWS SDK for Java 2.x da API.

Interromper uma instância

O exemplo de código a seguir mostra como interromper 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 [AWS Code Examples Repository](#).

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();
}
```



```
System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
ec2.stopInstances(request);
DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Successfully stopped instance " + instanceId);
}
```

- Para obter detalhes da API, consulte [StopInstances](#) na 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 [AWS Code Examples Repository](#).

```
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);
    }
}
```

- 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 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 saúde atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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("\n0kay, 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(terminateInstanceIRequest);
    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.getInstanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.getInstanceId().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.getInstanceProfile().getInstanceProfileName();
        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();

        iamClient.detachRolePolicy(detachPolicyRequest);
        System.out.println("Policy detached from entities.");
    }

    // Now, you can delete the policy.
    DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

    iamClient.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.getInstanceProfiles()) {
    RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
        .instanceProfileName(InstanceProfile)
        .roleName(roleName) // Remove the extra dot here
        .build();

    iamClient.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 da API do AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribeInstanceProfileAssociations](#)
 - [DescribeInstances](#)
 - [DescribeLoadBalancers](#)
 - [DescribeSubnets](#)
 - [DescribeTargetGroups](#)
 - [DescribeTargetHealth](#)
 - [DescribeVpcs](#)
 - [RebootInstances](#)

- [ReplacelamInstanceProfileAssociation](#)
- [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 = ""

            Usage:
                <keyName> <fileName> <groupName> <groupDesc> <vpcId>

            Where:
                keyName - A key pair name (for example, TestKeyPair).\s
                fileName - A file name where the key information is written to.
\s
                groupName - The name of the security group.\s
                groupDesc - The description of the security group.\s
                vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
                myIpAddress - The IP address of your development machine.\s

            """;

        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)
    .build();

SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .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("The instance type is " + instanceType);
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();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
        System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
        return instanceIdVal;

    } 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 {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .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());
            System.out.println("Instance type is " +
type.instanceType().toString());
            instanceType = type.instanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                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();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    }
}

```

```
    } 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 da 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 o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * 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 CreateCluster {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterName>\s

            Where:
                clusterName - The name of the ECS cluster to create.
            """;

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

        String clusterName = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        String clusterArn = createGivenCluster(ecsClient, clusterName);
        System.out.println("The cluster ARN is " + clusterArn);
        ecsClient.close();
    }

    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 do Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * 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 CreateService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName> <serviceName> <securityGroups>
<subnets> <taskDefinition>

                Where:
                clusterName - The name of the ECS cluster.
                serviceName - The name of the ECS service to
create.

                securityGroups - The name of the security group.
                subnets - The name of the subnet.
                taskDefinition - The name of the task definition.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceName = args[1];
        String securityGroups = args[2];
        String subnets = args[3];
        String taskDefinition = args[4];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
            taskDefinition);
        System.out.println("The ARN of the service is " + serviceArn);
        ecsClient.close();
    }
}
```

```
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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * 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 DeleteService {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <clusterName> <serviceArn>\s

                Where:
                clusterName - The name of the ECS cluster.
                serviceArn - The ARN of the ECS service.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String clusterName = args[0];
String serviceArn = args[1];
Region region = Region.US_EAST_1;
EcsClient ecsClient = EcsClient.builder()
    .region(region)
    .build();

deleteSpecificService(ecsClient, clusterName, serviceArn);
ecsClient.close();
}

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](#) a Referência AWS SDK for Java 2.x da API.

Descrever clusters

O exemplo de código a seguir mostra como descrever 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * 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 DescribeClusters {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <clusterArn> \s

            Where:
            clusterArn - The ARN of the ECS cluster to describe.
            """;

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

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        descCluster(ecsClient, clusterArn);
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Descrever 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;
```

```
/**
 * 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 ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <clusterArn> <taskId>\s

            Where:
                clusterArn - The ARN of an ECS cluster.
                taskId - The task Id value.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * 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 ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        listAllClusters(ecsClient);
        ecsClient.close();
    }

    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 do Amazon ECS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * 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 UpdateService {

    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <clusterName> <serviceArn>\s

            Where:
                clusterName - The cluster name.
                serviceArn - The service ARN value.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
            .region(region)
            .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }
}
```

```
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 do 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
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](#) a Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um receptor para o balanceador de carga (Load Balancer)

O exemplo de código a seguir mostra como criar um receptor que encaminha solicitações de um balanceador de carga do ELB para um grupo de destino.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
/*
 * 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 de destino ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
/*
 * 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;
    }
}
```

- Para obter detalhes da API, consulte [CreateTargetGroup](#) na Referência AWS SDK for Java 2.x da API.

Criar um Application Load Balancer

O exemplo de código a seguir mostra como criar um Application Load Balancer do ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
/*
 * 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 [CreateLoadBalancer](#) a Referência AWS SDK for Java 2.x da API.

Excluir um balanceador de carga

O exemplo de código a seguir mostra como excluir um balanceador de carga do ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
// 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 de destino do ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

// 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.

Obter a integridade de um grupo de destino

O exemplo de código a seguir mostra como obter a integridade de instâncias em um grupo de destino do ELB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
// 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 saúde atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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.deleteAutoScalingGroup(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 terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    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.getInstanceProfile().getInstanceProfileName();
        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.waitForTableExists(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 da API do AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)

- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

MediaStore 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 MediaStore.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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.

- [Ações](#)

Ações

Cria um contêiner

O exemplo de código a seguir mostra como criar um AWS Elemental MediaStore contêiner.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * 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 CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
        """;
```

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

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
                CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
                mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
                    containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
                containerResponse.container().arn());
            System.out.println("Finished ");

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

- Para obter detalhes da API, consulte [CreateContainer](#) a Referência AWS SDK for Java 2.x da API.

Excluir um contêiner

O exemplo de código a seguir mostra como excluir um AWS Elemental MediaStore contêiner.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * 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 CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to create.
            """;
```

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

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }

    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
        String containerName) {
        try {
            CreateContainerRequest containerRequest =
                CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
                mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
                    containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
                containerResponse.container().arn());
            System.out.println("Finished ");

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

- Para obter detalhes da API, consulte [DeleteContainer](#) a Referência AWS SDK for Java 2.x da API.

Excluir um objeto

O exemplo de código a seguir mostra como excluir um AWS Elemental MediaStore objeto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * 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 DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:    <completePath> <containerName>

                Where:
```

```
        completePath - The path (including the container) of the item to
delete.
        containerName - The name of the container.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String completePath = args[0];
    String containerName = args[1];
    Region region = Region.US_EAST_1;
    URI uri = new URI(getEndpoint(containerName));

    MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
        .endpointOverride(uri)
        .region(region)
        .build();

    deleteMediaObject(mediaStoreData, completePath);
    mediaStoreData.close();
}

public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
    try {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .path(completePath)
            .build();

        mediaStoreData.deleteObject(deleteObjectRequest);

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

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();
```

```
        DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
            .containerName(containerName)
            .build();

        DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- Para obter detalhes da API, consulte [DeleteObject](#) a Referência AWS SDK for Java 2.x da API.

Descrever um contêiner

O exemplo de código a seguir mostra como descrever um AWS Elemental MediaStore contêiner.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * 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 DescribeContainer {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <containerName>

            Where:
                containerName - The name of the container to describe.
            "";

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

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
            containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
        containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
            DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

            DescribeContainerResponse containerResponse =
            mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
            containerResponse.container().name());
            System.out.println("The container ARN is " +
            containerResponse.container().arn());
            return containerResponse.container().status().toString();
        } catch (MediaStoreException e) {
```



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

- Para obter detalhes da API, consulte [DescribeContainer](#) na Referência AWS SDK for Java 2.x da API.

Obter um objeto

O exemplo de código a seguir mostra como obter um AWS Elemental MediaStore objeto.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * 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 GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

            Usage:    <completePath> <containerName> <savePath>

            Where:
                completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                containerName - The name of the container.
                savePath - The path on the local drive where the file is saved,
including the file name (for example, C:/AWS/myvid.mp4).
            """;

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

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }

    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest.builder()

```

```

        .path(completePath)
        .build();

        // Write out the data to a file.
        ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
        byte[] buffer = new byte[data.available()];
        data.read(buffer);

        File targetFile = new File(savePath);
        OutputStream outputStream = new FileOutputStream(targetFile);
        outputStream.write(buffer);
        System.out.println("The data was written to " + savePath);

    } catch (MediaStoreDataException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static String getEndpoint(String containerName) {
    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}

```

- Para obter detalhes da API, consulte [GetObject](#) Referência AWS SDK for Java 2.x da API.

Listar contêineres

O exemplo de código a seguir mostra como listar AWS Elemental MediaStore contêineres.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * 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 ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
            .region(region)
            .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();

```

```
        for (Container container : containers) {
            System.out.println("Container name is " + container.name());
        }

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

- Para obter detalhes da API, consulte [ListContainers](#)sa Referência AWS SDK for Java 2.x da API.

Colocar um objeto em um contêiner

O exemplo de código a seguir mostra como colocar um objeto em um AWS Elemental MediaStore contêiner.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * 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 PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = ""

                To run this example, supply the name of a container, a file location
                to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

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

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
            .endpointOverride(uri)
            .region(region)
            .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                .path(completePath)

```

```
        .contentType("video/mp4")
        .build();

        PutObjectResponse response = mediaStoreData.putObject(objectRequest,
requestBody);
        System.out.println("The saved object is " +
response.storageClass().toString());

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

public static String getEndpoint(String containerName) {

    Region region = Region.US_EAST_1;
    MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
        .region(region)
        .build();

    DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
        .containerName(containerName)
        .build();

    DescribeContainerResponse response =
mediaStoreClient.describeContainer(containerRequest);
    return response.container().endpoint();
}
}
```

- Para obter detalhes da API, consulte [PutObject](#) Referência AWS SDK for Java 2.x da API.

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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

/**
 * 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 CreateDomain {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainName>

            Where:
                domainName - The name of the domain to create.
            """;

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

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        createNewDomain(searchClient, domainName);
        System.out.println("Done");
    }

    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](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * 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 DeleteDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to delete.
                """;

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

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
                .region(region)
                .build();
    }
}
```

```
        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    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](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
```

```
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

import java.util.List;

/**
 * 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 ListDomainNames {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        listAllDomains(searchClient);
        System.out.println("Done");
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * 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 UpdateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to update.
    }
}
```

```
        """;

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

    String domainName = args[0];
    Region region = Region.US_EAST_1;
    OpenSearchClient searchClient = OpenSearchClient.builder()
        .region(region)
        .build();

    updateSpecificDomain(searchClient, domainName);
    System.out.println("Done");
}

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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```



```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
            .region(region)
            .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](#) na Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Adicionar destino

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 [AWS Code Examples Repository](#).

Adicione um tópico do Amazon SNS como um 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 destino 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](#) na 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 [AWS Code Examples Repository](#).

Criar uma regra agendada

```
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);
    }
}
```

Criar 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" +\n"  \"detail-type\": [\"Object Created\"],\n" +\n"  \"detail\": {\n" +\n"    \"bucket\": {\n" +\n"      \"name\": [\"\" + bucketName + \"\"]\n" +\n"    }\n" +\n"  }\n" +\n"}";\n\ntry {\n    PutRuleRequest ruleRequest = PutRuleRequest.builder()\n        .description("Created by using the AWS SDK for Java v2")\n        .name(eventRuleName)\n        .eventPattern(pattern)\n        .roleArn(roleArn)\n        .build();\n\n    PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);\n    System.out.println("The ARN of the new rule is " +\nruleResponse.ruleArn());\n\n    } catch (EventBridgeException e) {\n        System.err.println(e.awsErrorDetails().errorMessage());\n        System.exit(1);\n    }\n}
```

- Para obter detalhes da API, consulte [PutRule](#) 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 [AWS Code Examples Repository](#).

```
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 [AWS Code Examples Repository](#).

```
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](#) na Referência AWS SDK for Java 2.x da API.

Desabilitar 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 [AWS Code Examples Repository](#).

Desabilitar uma regra usando o nome da 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 [AWS Code Examples Repository](#).

Habilitar uma regra usando o nome da 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 destino

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 [AWS Code Examples Repository](#).

Listar todos os nomes das regras usando a destino.

```
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](#) a Referência AWS SDK for Java 2.x da API.

Listar Regras

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 [AWS Code Examples Repository](#).

Habilitar uma regra usando o nome da 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](#) na Referência AWS SDK for Java 2.x da API.

Listar destinos 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 [AWS Code Examples Repository](#).

Listar todos os destinos 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> targetsList = res.targets();
    for (Target target: targetsList) {
        System.out.println("Target ARN: "+target.arn());
    }
}
```

- Para obter detalhes da API, consulte [ListTargetsByRule](#) na Referência AWS SDK for Java 2.x da API.

Remover destinos 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 [AWS Code Examples Repository](#).

Remover todos os destinos 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](#) na 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 [AWS Code Examples Repository](#).

```
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](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Conceitos básicos de regras e destinos

O código de exemplo a seguir mostra como:

- Criar uma regra e adicione um destino a ela.
- Habilitar e desabilitar regras.
- Listar e atualizar regras e destinos.
- Enviar eventos e limpar 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 = ""

            Usage:
                <roleName> <bucketName> <topicName> <eventRuleName>

            Where:
                roleName - The name of the role to create.
                bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
                topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
                eventRuleName - The Amazon EventBridge rule name to create.
            """;

        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)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

Region regionGl = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(regionGl)
    .build();

SnsClient snsClient = SnsClient.builder()
    .region(region)
    .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 listObjects = 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\"]," +
        "\"detail-type\": [\"ExampleType\"]" +
        "}";

    PutRuleRequest request = PutRuleRequest.builder()
        .name(ruleName)
        .description("Custom test rule")
        .eventPattern(customEventsPattern)
        .build();

    eventBrClient.putRule(request);
}

// Update an Amazon S3 object created rule with a transform on the target.
public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
    String targetId = java.util.UUID.randomUUID().toString();
    Map<String, String> myMap = new HashMap<>();
    myMap.put("bucket", "$.detail.bucket.name");
```

```
myMap.put("time", "$.time");

InputTransformer inputTransformer = InputTransformer.builder()
    .inputTemplate("\Notification: an object was uploaded to bucket
<bucket> at <time>.\")
    .inputPathsMap(myMap)
    .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 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.getAbsolutePath());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putObj = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(fileName)
```

```
        .build());

        s3Client.putObject(putOb, 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> targetsList = res.targets();
    for (Target target: targetsList) {
        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 da API do AWS SDK for Java 2.x .
 - [DeleteRule](#)
 - [DescribeRule](#)
 - [DisableRule](#)
 - [EnableRule](#)
 - [ListRuleNamesByTarget](#)
 - [ListRules](#)
 - [ListTargetsByRule](#)
 - [PutEvents](#)
 - [PutRule](#)
 - [PutTargets](#)

Exemplos do Forecast usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 conjunto de dados

O exemplo de código a seguir mostra como criar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * 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 CreateDataSet {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <name>\s

                Where:
                name - The name of the data set.\s
        """;
```

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

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
    }

    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 [CreateDataseta](#) Referência AWS SDK for Java 2.x da API.

Criar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * 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 CreateForecast {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <name> <predictorArn>\s

            Where:
                name - The name of the forecast.\s
                predictorArn - The arn of the predictor to use.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }
}
```

```
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 [CreateForecasta](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
```

```
* 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 DeleteDataset {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <datasetARN>\s

            Where:
                datasetARN - The ARN of the data set to delete.\s
            """;

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

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    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](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.forecast.ForecastClient;  
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;  
import software.amazon.awssdk.services.forecast.model.ForecastException;  
  
/**  
 * 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 DeleteDataset {  
  
    public static void main(String[] args) {  
        final String usage = ""  
  
        Usage:
```

```

        <datasetARN>\s

        Where:
            datasetARN - The ARN of the data set to delete.\s
        """;

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

    String datasetARN = args[0];
    Region region = Region.US_WEST_2;
    ForecastClient forecast = ForecastClient.builder()
        .region(region)
        .build();

    deleteForecastDataSet(forecast, datasetARN);
    forecast.close();
}

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 [DeleteForecasta](#) Referência AWS SDK for Java 2.x da API.

Descrever 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * 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 DescribeForecast {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <forecastarn>\s

                Where:
                forecastarn - The arn of the forecast (for example,
                "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

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

```
String forecastarn = args[0];
Region region = Region.US_WEST_2;
ForecastClient forecast = ForecastClient.builder()
    .region(region)
    .build();

describe(forecast, forecastarn);
forecast.close();
}

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 os 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * 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 ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
            .region(region)
            .build();

        listDataGroups(forecast);
        forecast.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Listar previsões

O exemplo de código a seguir mostra como listar previsões do Forecast.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.forecast.ForecastClient;  
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;  
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;  
import software.amazon.awssdk.services.forecast.model.ForecastSummary;  
import software.amazon.awssdk.services.forecast.model.ForecastException;  
import java.util.List;  
  
/**  
 * 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 ListForecasts {  
  
    public static void main(String[] args) {  
        Region region = Region.US_WEST_2;  
        ForecastClient forecast = ForecastClient.builder()
```

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

    listAllForeCasts(forecast);
    forecast.close();
}

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 Glue 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 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
            .maxResults(10)
            .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

```
}  
}
```

- Para obter detalhes da API, consulte [ListJobs](#) a Referência AWS SDK for Java 2.x da API.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.glue.GlueClient;  
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;  
import software.amazon.awssdk.services.glue.model.CrawlerTargets;  
import software.amazon.awssdk.services.glue.model.GlueException;  
import software.amazon.awssdk.services.glue.model.S3Target;  
import java.util.ArrayList;  
import java.util.List;  
  
/**  
 * 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 CreateCrawler {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <IAM> <s3Path> <cron> <dbName> <crawlerName>

            Where:
                IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
            """;

        if (args.length != 5) {
            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];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        glueClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * 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 GetCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

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

```

String crawlerName = args[0];
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

getSpecificCrawler(glueClient, crawlerName);
glueClient.close();
}

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](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * 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 GetDatabase {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <databaseName>

                Where:
                databaseName - The name of the database.\s
                """;

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

```
String databaseName = args[0];
Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();

getSpecificDatabase(glueClient, databaseName);
glueClient.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * 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 GetTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbName> <tableName>

            Where:
                dbName - The database name.\s
                tableName - The name of the table.\s
    }
}
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbName = args[0];
    String tableName = args[1];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getGlueTable(glueClient, dbName, tableName);
    glueClient.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * 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 StartCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
```



```
        System.out.println(usage);
        System.exit(1);
    }

    String crawlerName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    startSpecificCrawler(glueClient, crawlerName);
    glueClient.close();
}

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](#) 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.
- Liste informações sobre bancos de dados e tabelas em seu 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: Introdução ao 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 = ""

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                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)
    .build();
System.out.println(DASHES);
System.out.println("Welcome to the AWS Glue scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a database.");
createDatabase(glueClient, dbName, locationUri);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a crawler.");
createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait 5 min for the tables to become available");
TimeUnit.MINUTES.sleep(5);
System.out.println("6. Get tables.");
String myTableName = getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
```

```
        .build();

        CreateDatabaseRequest request = CreateDatabaseRequest.builder()
            .databaseInput(input)
            .build();

        glueClient.createDatabase(request);
        System.out.println(dbName + " was successfully created");

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

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");
    }
}
```

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

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        boolean ready = false;
        while (!ready) {
            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            String status = response.crawler().stateAsString();
            if (status.compareTo("READY") == 0) {
                ready = true;
            }
            Thread.sleep(3000);
        }

        System.out.println("The crawler is now ready");

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

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully started!");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

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

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

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

public static String getGlueTables(GlueClient glueClient, String dbName) {
    String myTableName = "";
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        if (tables.isEmpty()) {
            System.out.println("No tables were returned");
        } else {
            for (Table table : tables) {
                myTableName = table.name();
                System.out.println("Table name is: " + myTableName);
            }
        }
    }
}
```



```

        }
    }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return myTableName;
}

public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
    String outBucket) {
    try {
        Map<String, String> myMap = new HashMap<>();
        myMap.put("--input_database", inputDatabase);
        myMap.put("--input_table", inputTable);
        myMap.put("--output_bucket_url", outBucket);

        StartJobRunRequest runRequest = StartJobRunRequest.builder()
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .arguments(myMap)
            .jobName(jobName)
            .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

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

public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("glueetl")
            .scriptLocation(scriptLocation)
            .build();
    }
}

```

```
        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
            .role(iam)
            .command(command)
            .build();

        glueClient.createJob(jobRequest);
        System.out.println(jobName + " was successfully created.");

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

public static void getAllJobs(GlueClient glueClient) {
    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job : jobs) {
            System.out.println("Job name is : " + job.name());
            System.out.println("The job worker type is : " +
job.workerType().name());
        }

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

public static void getJobRuns(GlueClient glueClient, String jobName) {
    try {
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
            .jobName(jobName)
```

```

        .maxResults(20)
        .build();

    boolean jobDone = false;
    while (!jobDone) {
        GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
        List<JobRun> jobRuns = response.jobRuns();
        for (JobRun jobRun : jobRuns) {
            String jobState = jobRun.jobRunState().name();
            if (jobState.compareTo("SUCCEEDED") == 0) {
                System.out.println(jobName + " has succeeded");
                jobDone = true;

            } else if (jobState.compareTo("STOPPED") == 0) {
                System.out.println("Job run has stopped");
                jobDone = true;

            } else if (jobState.compareTo("FAILED") == 0) {
                System.out.println("Job run has failed");
                jobDone = true;

            } else if (jobState.compareTo("TIMEOUT") == 0) {
                System.out.println("Job run has timed out");
                jobDone = true;

            } else {
                System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                System.out.println("Job run Id is " + jobRun.id());
                System.out.println("The Glue version is " +
jobRun.glueVersion());
            }
            TimeUnit.SECONDS.sleep(5);
        }
    }

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

public static void deleteJob(GlueClient glueClient, String jobName) {
    try {

```

```
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

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

public static void deleteDatabase(GlueClient glueClient, String databaseName) {
    try {
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
            .name(databaseName)
            .build();

        glueClient.deleteDatabase(request);
        System.out.println(databaseName + " was successfully deleted");

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

public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.deleteCrawler(deleteCrawlerRequest);
        System.out.println(crawlerName + " was deleted");

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

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Copiar um conjunto de imagens

O exemplo de código a seguir mostra como copiar um conjunto de HealthImaging imagens.

SDK para Java 2.x

```
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](#) a Referência AWS SDK for Java 2.x da API.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Criar um datastore

O exemplo de código a seguir mostra como criar um armazenamento HealthImaging de dados.

SDK para Java 2.x

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Excluir um datastore

O exemplo de código a seguir mostra como excluir um armazenamento HealthImaging de dados.

SDK para Java 2.x

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Excluir um conjunto de imagens

O exemplo de código a seguir mostra como excluir um conjunto de HealthImaging imagens.

SDK para Java 2.x

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obter um quadro de imagem

O exemplo de código a seguir mostra como obter um quadro de imagem.

SDK para Java 2.x

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obter propriedades do datastore

O exemplo de código a seguir mostra como obter propriedades do armazenamento de HealthImaging dados.

SDK para Java 2.x

```
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](#) Referência AWS SDK for Java 2.x da API.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obter propriedades do trabalho de importação

O exemplo de código a seguir mostra como obter as propriedades do trabalho de importação.

SDK para Java 2.x

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obter metadados para um conjunto de imagem

O exemplo de código a seguir mostra como obter metadados para um conjunto de HealthImaging imagens.

SDK para Java 2.x

```
public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
    String destinationPath,
    String datastoreId,
    String imagesetId,
    String versionId) {


    try {
        GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
        GetImageSetMetadataRequest.builder()
            .datastoreId(datastoreId)
            .imageSetId(imagesetId);

        if (versionId != null) {
            getImageSetMetadataRequestBuilder =
            getImageSetMetadataRequestBuilder.versionId(versionId);
        }

        medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
            FileSystems.getDefault().getPath(destinationPath));

        System.out.println("Metadata downloaded to " + destinationPath);
    } catch (MedicalImagingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [GetImageSetMetadata](#) a Referência AWS SDK for Java 2.x da API.

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Importar dados em massa para um datastore

O exemplo de código a seguir mostra como importar dados em massa para um armazenamento HealthImaging de dados.

SDK para Java 2.x

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Listar datastores

O exemplo de código a seguir mostra como listar armazenamentos HealthImaging de dados.

SDK para Java 2.x

```
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](#) na Referência AWS SDK for Java 2.x da API.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Listar trabalhos de importação para um datastore

O exemplo de código a seguir mostra como listar trabalhos de importação para um armazenamento HealthImaging de dados.

SDK para Java 2.x

```
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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Listar as tags para um recurso

O exemplo de código a seguir mostra como listar as tags de um HealthImaging recurso.


SDK para Java 2.x

```
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](#) a Referência AWS SDK for Java 2.x da API.

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Remover tag de um recurso

O exemplo de código a seguir mostra como remover uma tag de um HealthImaging recurso.

SDK para Java 2.x

```
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](#) Referência AWS SDK for Java 2.x da API.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Pesquisar conjuntos de imagens

O exemplo de código a seguir mostra como pesquisar conjuntos de HealthImaging imagens.

SDK para Java 2.x

A função de utilitário 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 nº 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 nº 3: operador BETWEEN usando createdAt. Os estudos de tempo foram previamente persistidos.

```

        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](#) na Referência AWS SDK for Java 2.x da API.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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

```

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.

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Cenários

Marcar um datastore

O exemplo de código a seguir mostra como marcar um armazenamento HealthImaging de dados.

SDK para Java 2.x

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 tags para 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 de utilitário 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 da API do AWS SDK for Java 2.x .
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Marcar um conjunto de imagens

O exemplo de código a seguir mostra como marcar um conjunto de HealthImaging imagens.

SDK para Java 2.x

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 de utilitário 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 da API do AWS SDK for Java 2.x .
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Exemplos do IAM usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * 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 HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```



```
}
```

- Para obter detalhes da API, consulte [ListPolicies](#) a Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Anexar uma política a uma função

O exemplo de código a seguir mostra como anexar uma política do IAM a um perfil.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleName> <policyArn>\s

            Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    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](#) 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreatePolicy {

    public static final String PolicyDocument = "{" +
        "  \"Version\": \"2012-10-17\",\" +
        "  \"Statement\": [\" +
        "    {\" +
        "      \"Effect\": \"Allow\",\" +
        "      \"Action\": [\" +
        "        \"dynamodb:DeleteItem\",\" +
        "        \"dynamodb:GetItem\",\" +
        "        \"dynamodb:PutItem\",\" +
        "        \"dynamodb:Scan\",\" +
        "        \"dynamodb:UpdateItem\"" +
        "    ],\" +
        "    \"Resource\": \"*\":" +
        "  }" +
```

```
        "]" +
        "}";

public static void main(String[] args) {

    final String usage = ""
        Usage:
            CreatePolicy <policyName>\s

        Where:
            policyName - A unique policy name.\s
        """;

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

    String policyName = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String result = createIAMPolicy(iam, policyName);
    System.out.println("Successfully created a policy with this ARN value: " +
result);
    iam.close();
}

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 um perfil

O exemplo de código a seguir mostra como criar um perfil do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*

```

```
* This example requires a trust policy document. For more information, see:
* https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
*
* In addition, 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 CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = ""
            Usage:
                <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
the trust policy.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
```

```
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is " + response.role().arn());

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

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
}
```

- Para obter detalhes da API, consulte [CreateRole](#) na 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreateUser {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <username>\s

                Where:
                username - The name of the user to create.\s
                """;

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

```
String username = args[0];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();

String result = createIAMUser(iam, username);
System.out.println("Successfully created user: " + result);
iam.close();
}

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 "";
}
}
```

- Para obter detalhes da API, consulte [CreateUser](#) a 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreateAccessKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <user>\s

                Where:
```

```
        user - An AWS IAM user that you can obtain from the AWS
Management Console.
        """;

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

    String user = args[0];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    String keyId = createIAMAccessKey(iam, user);
    System.out.println("The Key Id is " + keyId);
    iam.close();
}

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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <alias>\s

            Where:
                alias - The account alias to create (for example, myawsaccount).

\s

        """;

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

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
```

```
IamClient iam = IamClient.builder()
    .region(region)
    .build();

createIAMAccountAlias(iam, alias);
iam.close();
System.out.println("Done");
}

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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 DeletePolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <policyARN>\s

            Where:
                policyARN - A policy ARN value to delete.\s
            """;

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

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Excluir 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DeleteUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <userName>\s

            Where:
                userName - The name of the user to delete.\s
            """;

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

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    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](#) 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 [AWS Code Examples Repository](#).

```
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;  
  
/**  
 * 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 DeleteAccessKey {  
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <username> <accessKey>\s

    Where:
        username - The name of the user.\s
        accessKey - The access key ID for the secret access key you want
to delete.\s

    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String username = args[0];
String accessKey = args[1];
Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .build();
deleteKey(iam, username, accessKey);
iam.close();
}

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](#) a 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 da conta do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <alias>\s

            Where:
                alias - The account alias to delete.\s
        """;
```

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

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Desanexar uma política de uma função

O exemplo de código a seguir mostra como desanexar 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <roleName> <policyArn>\s

                Where:
                roleName - A role name that you can obtain from the AWS
Management Console.\s
                policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
```

```
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();
    detachPolicy(iam, roleName, policyArn);
    System.out.println("Done");
    iam.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 ListAccessKeys {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <userName>\s

                Where:
                userName - The name of the user for which access keys are
retrieved.\s

                """;

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

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
```



```
IamClient iam = IamClient.builder()
    .region(region)
    .build();

listKeys(iam, userName);
System.out.println("Done");
iam.close();
}

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 conta do IAM.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
```

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

    listAliases(iam);
    System.out.println("Done");
    iam.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
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.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * 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 ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(region)
            .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
```

```
        if (newMarker == null) {
            ListUsersRequest request = ListUsersRequest.builder().build();
            response = iam.listUsers(request);
        } else {
            ListUsersRequest request = ListUsersRequest.builder()
                .marker(newMarker)
                .build();

            response = iam.listUsers(request);
        }

        for (User user : response.users()) {
            System.out.format("\n Retrieved user %s", user.userName());
            AttachedPermissionsBoundary permissionsBoundary =
user.permissionsBoundary();
            if (permissionsBoundary != null)
                System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
        }

        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

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

- Para obter detalhes da API, consulte [ListUsers](#) na 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

ℹ Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 UpdateUser {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <curName> <newName>\s

            Where:
                curName - The current user name.\s
                newName - An updated user name.\s

            """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String curName = args[0];
    String newName = args[1];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateIAMUser(iam, curName, newName);
    System.out.println("Done");
    iam.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <username> <accessId> <status>\s

            Where:
                username - The name of the user whose key you want to update.\s
    }
```



```

        accessId - The access key ID of the secret access key you want
to update.\s
        status - The status you want to assign to the secret access key.
\s
        """;

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

    String username = args[0];
    String accessId = args[1];
    String status = args[2];
    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .build();

    updateKey(iam, username, accessId, status);
    System.out.println("Done");
    iam.close();
}

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 [UpdateAccessKey](#) 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 saúde atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

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 =
HttpClient.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("\n0kay, 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 terminateInstanceIRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    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
    .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.getInstanceProfile().getInstanceProfileName();
        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 da API do AWS SDK for Java 2.x .
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribeIamInstanceProfileAssociations](#)
 - [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 um perfil.

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 = ""

            Usage:
              <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

            Where:
              username - The name of the IAM user to create.\s
              policyName - The name of the policy to create.\s
              roleName - The name of the role to create.\s
              roleSessionName - The name of the session required for the
assumeRole operation.\s
              bucketName - The name of the Amazon S3 bucket from which objects
are read.\s

            """;

        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)
        .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 da 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 [AWS Code Examples Repository](#).

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();
        // 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 da API do AWS SDK for Java 2.x .
 - [CreatePolicy](#)
 - [GetPolicy](#)
 - [GetPolicyVersion](#)

AWS IoT 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 IoT.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();

        ListThingsResponse response = iotClient.listThings(thingsRequest) ;
        List<ThingAttribute> thingList = response.things();
        for (ThingAttribute attribute : thingList) {
```

```
        System.out.println("Thing name: "+attribute.thingName());
        System.out.println("Thing ARN: "+attribute.thingArn());
    }
}
}
```

- Para obter detalhes da API, consulte [ListThings](#) na Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Anexar um certificado

O exemplo de código a seguir mostra como anexar um AWS IoT certificado.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
        .thingName(thingName)
        .principal(certificateArn)
        .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
```

```
        if (attachResponse.sdkHttpResponse().isSuccessful()) {
            System.out.println("Certificate attached to Thing successfully.");

            // Print additional information about the Thing.
            describeThing(iotClient, thingName);
        } else {
            System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
                attachResponse.sdkHttpResponse().statusCode());
        }
    }
}
```

- Para obter detalhes da API, consulte [AttachThingPrincipal](#) na Referência AWS SDK for Java 2.x da API.

Criar um certificado

O exemplo de código a seguir mostra como criar um AWS IoT certificado.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;
    }
}
```



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

    return "";
}
```

- Para obter detalhes da API, consulte [CreateKeysAndCertificate](#) a Referência AWS SDK for Java 2.x da API.

Criar uma regra

O exemplo de código a seguir mostra como criar uma AWS IoT regra.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
```

```
        .sql(sql)
        .actions(myAction)
        .build();

    // Create the topic rule request.
    CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
        .ruleName(ruleName)
        .topicRulePayload(topicRulePayload)
        .build();

    // Create the rule.
    iotClient.createTopicRule(topicRuleRequest);
    System.out.println("IoT Rule created successfully.");

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

- Para obter detalhes da API, consulte [CreateTopicRule](#) Referência AWS SDK for Java 2.x da API.

Criar um objeto

O exemplo de código a seguir mostra como criar AWS IoT algo.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();
```

```
        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

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

- Para obter detalhes da API, consulte [CreateThing](#) Referência AWS SDK for Java 2.x da API.

Excluir um certificado

O exemplo de código a seguir mostra como excluir um AWS IoT certificado.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

- Para obter detalhes da API, consulte [DeleteCertificate](#) Referência AWS SDK for Java 2.x da API.

Excluir um objeto

O exemplo de código a seguir mostra como excluir AWS IoT algo.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [DeleteThing](#) Referência AWS SDK for Java 2.x da API.

Descreva uma coisa

O exemplo de código a seguir mostra como descrever qualquer AWS IoT coisa.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

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

- Para obter detalhes da API, consulte [DescribeThing](#) Referência AWS SDK for Java 2.x da API.

Desanexar um certificado

O exemplo de código a seguir mostra como desanexar um AWS IoT certificado.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
```

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

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn +" was successfully removed from "
+thingName);


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

- Para obter detalhes da API, consulte [DetachThingPrincipal](#) a Referência AWS SDK for Java 2.x da API.

Obter informações do endpoint

O exemplo de código a seguir mostra como obter informações AWS IoT do endpoint.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
        iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;
    }
}
```

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

- Para obter detalhes da API, consulte [DescribeEndpoint](#) Referência AWS SDK for Java 2.x da API.

Liste seus certificados

O exemplo de código a seguir mostra como listar seus AWS IoT certificados.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).


```
public static void listCertificates(IotClient iotClient) {  
    ListCertificatesResponse response = iotClient.listCertificates();  
    List<Certificate> certList = response.certificates();  
    for (Certificate cert : certList) {  
        System.out.println("Cert id: " + cert.certificateId());  
        System.out.println("Cert Arn: " + cert.certificateArn());  
    }  
}
```

- Para obter detalhes da API, consulte [ListCertificates](#) Referência AWS SDK for Java 2.x da API.

Consulte o índice de pesquisa

O exemplo de código a seguir mostra como consultar o índice AWS IoT de pesquisa.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
        iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
            System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [SearchIndex](#) Referência AWS SDK for Java 2.x da API.

Atualizar um objeto

O exemplo de código a seguir mostra como atualizar qualquer AWS IoT coisa.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [UpdateThing](#) a Referência AWS SDK for Java 2.x da API.

Cenários

Trabalhe com casos de uso de gerenciamento de dispositivos

O exemplo de código a seguir mostra como trabalhar com casos de uso de gerenciamento de AWS IoT dispositivos usando o AWS IoT SDK

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
```

```
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * 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 an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach and delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String TOPIC = "your-iot-topic";
    public static void main(String[] args) {
        final String usage =
            ""
```

```
Usage:
    <roleARN> <snsAction>

Where:
    roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
    snsAction - An ARN of an SNS topic.
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String thingName;
String ruleName;
String roleARN = args[0];
String snsAction = args[1];
Scanner scanner = new Scanner(System.in);
IoTClient iotClient = IoTClient.builder()
    .region(Region.US_EAST_1)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IoT example workflow.");
System.out.println("""
    This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series of
steps,
    including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
    It utilizes the AWS SDK for Java V2 and incorporates functionality for
creating and managing IoT Things, certificates, rules,
    shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
    developers working with AWS IoT in a Java environment.

""");
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an AWS IoT Thing.");
```

```
System.out.println("""
    An AWS IoT Thing represents a virtual entity in the AWS IoT service that
    can be associated with a physical device.
    """);
// Prompt the user for input.
System.out.print("Enter Thing name: ");
thingName = scanner.nextLine();
createIoTThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Generate a device certificate.");
System.out.println("""
    A device certificate performs a role in securing the communication
    between devices (Things) and the AWS IoT platform.
    """);

System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
String certAns = scanner.nextLine();
String certificateArn="" ;
if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
    certificateArn = createCertificate(iotClient);
    System.out.println("Attach the certificate to the AWS IoT Thing.");
    attachCertificateToThing(iotClient, thingName, certificateArn);
} else {
    System.out.println("A device certificate was not created.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Update an AWS IoT Thing with Attributes.");
System.out.println("""
    IoT Thing attributes, represented as key-value pairs, offer a pivotal
    advantage in facilitating efficient data
    management and retrieval within the AWS IoT ecosystem.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
updateThing(iotClient, thingName);
System.out.println(DASHES);

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

```
System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
System.out.println("""
    An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
String endpointUrl = describeEndpoint(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List your AWS IoT certificates");
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
    listCertificates(iotClient);
} else {
    System.out.println("You did not create a certificates. Skipping this
step.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
System.out.println("""
    A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
    of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
    the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
    .region(Region.US_EAST_1)
    .endpointOverride(URI.create(endpointUrl))
    .build();

updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("7. Write out the state information, in JSON format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Creates a rule");
System.out.println("""
Creates a rule that is an administrator-level action.
Any user who has permission to create rules will be able to access data
processed by the rule.
""");
System.out.print("Enter Rule name: ");
ruleName = scanner.nextLine();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
System.out.println(DASHES);

System.out.println(DASHES);
if (certificateArn.length() > 0) {
    System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        System.out.println("11. You selected to detach amd delete the
certificate.");
        System.out.print("Press Enter to continue...");
```

```
        scanner.nextLine();
        detachThingPrincipal(iotClient, thingName, certificateArn);
        deleteCertificate(iotClient, certificateArn);
    } else {
        System.out.println("11. You selected not to delete the
certificate.");
    }
    } else {
        System.out.println("11. You did not create a certificate so there is
nothing to delete.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("12. Delete the AWS IoT Thing.");
    System.out.print("Do you want to delete the IoT Thing? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        deleteIoTThing(iotClient, thingName);
    } else {
        System.out.println("The IoT Thing was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The AWS IoT workflow has successfully completed.");
    System.out.println(DASHES);
}

public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}

public static void listIoTRules(IotClient iotClient) {
    try {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
    }
```



```
        System.out.println("List of IoT Rules:");
        List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
        for (TopicRuleListItem rule : ruleList) {
            System.out.println("Rule Name: " + rule.ruleName());
            System.out.println("Rule ARN: " + rule.ruleArn());
            System.out.println("-----");
        }

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

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");
    }
}
```

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

public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

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

public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
```

```
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

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

public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");

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

public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
        iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
```

```
        String endpointUrl = endpointResponse.endpointAddress();
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

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

public static void detachThingPrincipal(IotClient iotClient, String thingName,
String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

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

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

```
// Get the cert Id from the Cert ARN value.
private static String extractCertificateId(String certificateArn) {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    String[] arnParts = certificateArn.split(":");
    String certificateIdPart = arnParts[arnParts.length - 1];
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
}

public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

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

    return "";
}

public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
        .thingName(thingName)
        .principal(certificateArn)
        .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
```

```
        if (attachResponse.sdkHttpResponse().isSuccessful()) {
            System.out.println("Certificate attached to Thing successfully.");

            // Print additional information about the Thing.
            describeThing(iotClient, thingName);
        } else {
            System.err.println("Failed to attach certificate to Thing. HTTP Status
Code: " +
                attachResponse.sdkHttpResponse().statusCode());
        }
    }

    private static void describeThing(IotClient iotClient, String thingName) {
        try {
            DescribeThingRequest thingRequest = DescribeThingRequest.builder()
                .thingName(thingName)
                .build();

            // Print Thing details.
            DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
            System.out.println("Thing Details:");
            System.out.println("Thing Name: " + describeResponse.thingName());
            System.out.println("Thing ARN: " + describeResponse.thingArn());

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

    public static void deleteIoTThing(IotClient iotClient, String thingName) {
        try {
            DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
                .thingName(thingName)
                .build();

            iotClient.deleteThing(deleteThingRequest);
            System.out.println("Deleted Thing " + thingName);

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

```
}

public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN value
is " + createThingResponse.thingArn());

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

private static String getValue(String input) {
    // Define a regular expression pattern for extracting the subdomain.
    Pattern pattern = Pattern.compile("^(.*)\\.\\.iot\\.\\.us-east-1\\.\\.amazonaws\\.
\\.com");

    // Match the pattern against the input string.
    Matcher matcher = pattern.matcher(input);

    // Check if a match is found.
    if (matcher.find()) {
        // Extract the subdomain from the first capturing group.
        String subdomain = matcher.group(1);
        System.out.println("Extracted subdomain: " + subdomain);
        return subdomain ;
    } else {
        System.out.println("No match found");
    }
    return "" ;
}

public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();
```

```
    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

AWS IoT data 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 IoT data.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Pegue a sombra

O exemplo de código a seguir mostra como obter a sombra de uma AWS IoT coisa.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

        // Extracting payload from response.
        SdkBytes payload = getThingShadowResponse.payload();
        String payloadString = payload.asUtf8String();
        System.out.println("Received Shadow Data: " + payloadString);

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

- Para obter detalhes da API, consulte [GetThingShadow](#) a Referência AWS SDK for Java 2.x da API.

Atualize a sombra

O exemplo de código a seguir mostra como atualizar a sombra de qualquer AWS IoT coisa.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void updateShadowThing(IotDataPlaneClient iotPlaneClient, String
thingName) {
    try {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
        \"humidity\":50}}}\"";
        SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        // Update Thing Shadow.
        iotPlaneClient.updateThingShadow(updateThingShadowRequest);
        System.out.println("Thing Shadow updated successfully.");

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

- Para obter detalhes da API, consulte [UpdateThingShadow](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Keyspaces usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * 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)
            .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](#) na Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar um espaço de chaves

O exemplo de código a seguir mostra como criar um espaço de chave do Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

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

- 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 [AWS Code Examples Repository](#).

```
public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
            .name("release_date")
            .type("timestamp")
            .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
            .name("plot")
            .type("text")
            .build();

        List<ColumnDefinition> collist = new ArrayList<>();
        collist.add(defTitle);
        collist.add(defYear);
        collist.add(defReleaseDate);
        collist.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();
```

```
PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

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

- 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 espaço de chaves do Amazon Keyspaces.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);


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

- 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 [AWS Code Examples Repository](#).


```
public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

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

- Para obter detalhes da API, consulte [DeleteTable](#) a Referência AWS SDK for Java 2.x da API.

Obter 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 [AWS Code Examples Repository](#).

```
public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");
    }
}
```

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

- Para obter detalhes da API, consulte [GetKeyspace](#) 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 [AWS Code Examples Repository](#).

```
public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
        }
    }
}
```

```

        }
        Thread.sleep(500);
    }

    List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
    for (ColumnDefinition def : cols) {
        System.out.println("The column name is " + def.name());
        System.out.println("The column type is " + def.type());
    }

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

```

- Para obter detalhes da API, consulte [GetTable](#) a Referência AWS SDK for Java 2.x da API.

Listar espaços de chaves

O exemplo de código a seguir mostra como listar espaços de chaves do Amazon Keyspaces.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
            keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
    }
}

```

```

        .forEach(content -> System.out.println(" Name: " +
content.keyspaceName())));

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

```

- 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 espaço de chave.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {

```

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

- Para obter detalhes da API, consulte [ListTables](#) Referência AWS SDK for Java 2.x da API.

Restaurar uma tabela para um ponto no tempo

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 [AWS Code Examples Repository](#).

```
public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
    ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
            keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
            response.restoredTableARN());

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

```
}
```

- Para obter detalhes da API, consulte [RestoreTable](#) Referência AWS SDK for Java 2.x da API.

Atualizar uma tabela

O exemplo de código a seguir mostra como atualizar um item em 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 [AWS Code Examples Repository](#).

```
public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);

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

- Para obter detalhes da API, consulte [UpdateTable](#) Referência AWS SDK for Java 2.x da API.

Cenários

Começar a usar espaços de chaves e tabelas

O código de exemplo a seguir mostra como:

- Criar um keyspace e uma tabela. O esquema da tabela contém dados do filme e tem a point-in-time recuperação ativada.
- Conectar-se ao espaço de chaves usando uma conexão TLS segura com autenticação SigV4.
- Consulte a tabela. Adicionar recuperar e atualizar dados do filme.
- Atualizar tabela. Adicionar uma coluna para rastrear os filmes assistidos.
- Restaurar a tabela ao estado anterior e limpar 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)
            .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)
            .addColumnns(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> collList = new ArrayList<>();
        collList.add(defTitle);
        collList.add(defYear);
        collList.add(defReleaseDate);
        collList.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
            .name("year")
            .build();

        PartitionKey titleKey = PartitionKey.builder()
```

```
        .name("title")
        .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(colList)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
    .build();

CreateTableResponse response = keyClient.createTable(tableRequest);
System.out.println("The table ARN is " + response.resourceArn());

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

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
            .maxResults(10)
            .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));
    }
}
```

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

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

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

    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                .keyspaceName(keyspaceName)
                .build();

            CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API do AWS SDK for Java 2.x .
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

Exemplos do Kinesis usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * 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 CreateDataStream {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
                StockTradeStream).
            """;

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

```
    }

    String streamName = args[0];
    Region region = Region.US_EAST_1;
    KinesisClient kinesisClient = KinesisClient.builder()
        .region(region)
        .build();
    createStream(kinesisClient, streamName);
    System.out.println("Done");
    kinesisClient.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * 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 DeleteDataStream {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <streamName>

            Where:
                streamName - The Amazon Kinesis data stream (for example,
StockTradeStream)
            """;

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

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
            .region(region)
            .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }
}
```



```
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](#) Referência AWS SDK for Java 2.x da API.

Obter 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](#).

```
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.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
```

```
import java.util.ArrayList;
import java.util.List;

/**
 * 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 GetRecords {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <streamName>

                Where:
                streamName - The Amazon Kinesis data stream to read from (for
example, StockTradeStream).
                """;

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

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        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 da API do AWS SDK for Java 2.x .
 - [GetRecords](#)
 - [GetShardIterator](#)

Inserir dados em um stream

O exemplo de código a seguir mostra como inserir 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 [AWS Code Examples Repository](#).

```
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;  
  
/**  
 * 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 StockTradesWriter {  
    public static void main(String[] args) {  
        final String usage = ""
```

```

        Usage:
            <streamName>

        Where:
            streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
            """;

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

    String streamName = args[0];
    Region region = Region.US_EAST_1;
    KinesisClient kinesisClient = KinesisClient.builder()
        .region(region)
        .build();

    // Ensure that the Kinesis Stream is valid.
    validateStream(kinesisClient, streamName);
    setStockData(kinesisClient, streamName);
    kinesisClient.close();
}

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

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de fluxo do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

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 Kinesis com o Lambda usando Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
}
```

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 fluxo 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * 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 CreateGrant {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId> <granteePrincipal> <operation>\s

            Where:
                keyId - The unique identifier for the customer master key (CMK)
that the grant applies to.\s
                granteePrincipal - The principal that is given permission to
perform the operations that the grant permits.\s
                operation - An operation (for example, Encrypt).\s
            """;

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

        String keyId = args[0];
        String granteePrincipal = args[1];
        String operation = args[2];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        String grantId = createGrant(kmsClient, keyId, granteePrincipal, operation);
    }
}
```

```
        System.out.printf("Successfully created a grant with ID %s%n", grantId);
        kmsClient.close();
    }

    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.

Criar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
```

```
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * 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 CreateCustomerKey {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        String keyDesc = "Created by the AWS KMS API";
        System.out.println("The key id is " + createKey(kmsClient, keyDesc));
        kmsClient.close();
    }

    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](#) a Referência AWS SDK for Java 2.x da API.

Criar um alias para uma chave

O exemplo de código a seguir mostra como criar um alias para uma chave do KMS Key.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * 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 CreateAlias {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <targetKeyId> <aliasName>\s

            Where:
                targetKeyId - The key ID or the Amazon Resource Name (ARN) of
the customer master key (CMK).\s
                aliasName - An alias name (for example, alias/myAlias).\s
    }
}
```

```
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String targetKeyId = args[0];
    String aliasName = args[1];
    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    createCustomAlias(kmsClient, targetKeyId, aliasName);
    kmsClient.close();
}

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 com uma chave KMS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        decryptResponse.plaintext();


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

- Para obter detalhes da API, consulte [Decrypt](#) na Referência de API do AWS SDK for Java 2.x .

Descrever uma chave

O exemplo de código a seguir mostra como descrever uma chave do KMS.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).


```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * 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 DescribeKey {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <keyId>\s

            Where:
                keyId - A key id value to describe (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

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

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        describeSpecifcKey(kmsClient, keyId);
        kmsClient.close();
    }

    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](#) Referência AWS SDK for Java 2.x da API.

Desabilitar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.KmsException;

/**
 * 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 DisableCustomerKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - A key id value to disable (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

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

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        disableKey(kmsClient, keyId);
        kmsClient.close();
    }

    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](#) Referência AWS SDK for Java 2.x da API.

Habilitar uma chave

O exemplo de código a seguir mostra como habilitar uma chave do KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;

/**
 * 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 EnableCustomerKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - A key id value to enable (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String keyId = args[0];
    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    enableKey(kmsClient, keyId);
    kmsClient.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;

/**
 * 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 EncryptDataKey {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - A key id value to use to encrypt/decrypt the data (for
                example, xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s

                """;

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

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        SdkBytes encryData = encryptData(kmsClient, keyId);
        decryptData(kmsClient, encryData, keyId);
    }
}
```

```
        System.out.println("Done");
        kmsClient.close();
    }

    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;
    }

    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 [Encrypt](#) na Referência de API do AWS SDK for Java 2.x .

Listar apelidos para uma chave

O exemplo de código a seguir mostra como listar aliases para uma chave do KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListAliasesResponse;
import java.util.List;

/**
 * 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 ListAliases {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllAliases(kmsClient);
        kmsClient.close();
    }
}
```



```
}

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 concessões para uma chave

O exemplo de código a seguir mostra como listar concessões para uma chave do KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.GrantListEntry;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
```

```
import software.amazon.awssdk.services.kms.model.ListGrantsResponse;
import java.util.List;

/**
 * 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 ListGrants {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <keyId>\s

                Where:
                keyId - a key id value to use (for example,
                xxxxxbcd-12ab-34cd-56ef-1234567890ab).\s
                """;

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

        String keyId = args[0];
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        displayGrantIds(kmsClient, keyId);
        kmsClient.close();
    }

    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](#) a Referência AWS SDK for Java 2.x da API.

Listar chaves

O exemplo de código a seguir mostra como listar chaves do KMS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.KeyListEntry;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.ListKeysResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import java.util.List;

/**
 * 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 ListKeys {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    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 do Lambda usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 Lambda

Os exemplos de código a seguir mostram como começar a usar o Lambda.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
package com.example.lambda;

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;

/**
 * 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 ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }

    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);
        }
    }
}
```

- Para obter detalhes da API, consulte [ListFunctions](#) na Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

Ações

Criar uma função

O exemplo de código a seguir mostra como criar uma função do Lambda.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;
import software.amazon.awssdk.services.lambda.model.FunctionCode;
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *
 * https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/
creating_workflows_stepfunctions
 *
 * 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 CreateFunction {
    public static void main(String[] args) {
```

```

final String usage = ""

    Usage:
        <functionName> <filePath> <role> <handler>\s

    Where:
        functionName - The name of the Lambda function.\s
        filePath - The path to the ZIP or JAR where the code is located.
\s
        role - The role ARN that has Lambda permissions.\s
        handler - The fully qualified method name (for example,
example.Handler::handleRequest). \s
        """;

if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String functionName = args[0];
String filePath = args[1];
String role = args[2];
String handler = args[3];
Region region = Region.US_WEST_2;
LambdaClient awsLambda = LambdaClient.builder()
    .region(region)
    .build();

createLambdaFunction(awsLambda, functionName, filePath, role, handler);
awsLambda.close();
}

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](#) Referência AWS SDK for Java 2.x da API.

Excluir uma função

O exemplo de código a seguir mostra como excluir uma função do Lambda.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DeleteFunction {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function.\s
            """;

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

        String functionName = args[0];
        Region region = Region.US_EAST_1;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();
```

```
        deleteLambdaFunction(awsLambda, functionName);
        awsLambda.close();
    }

    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](#) a Referência AWS SDK for Java 2.x da API.

Invocar uma função

O exemplo de código a seguir mostra como invocar uma função do Lambda.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
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;

public class LambdaInvoke {

    /*
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * 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 static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function\s
            """;

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

        String functionName = args[0];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        invokeFunction(awsLambda, functionName);
        awsLambda.close();
    }
}
```

```
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 da 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 = ""
```

Usage:

```
<functionName> <filePath> <role> <handler> <bucketName> <key>\s
```

Where:

```
functionName - The name of the Lambda function.\s
```

```
filePath - The path to the .zip or .jar where the code is
located.\s
```

```
role - The AWS Identity and Access Management (IAM) service role
that has Lambda permissions.\s
```

```
handler - The fully qualified method name (for example,
example.Handler::handleRequest).\s
```

```
bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name that contains the .zip or .jar used to update the Lambda function's
code.\s
```

```
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)
    .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 da 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 trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de fluxo do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

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 Kinesis com o Lambda usando Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
        }
    }
}
```

```
    }
    catch (Exception ex) {
        logger.log("An error occurred:"+ex.getMessage());
        throw ex;
    }
}
logger.log("Successfully processed:"+event.getRecords().size()+" records");
return null;
}
}
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do 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);
    }
}
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

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 SNS com o Lambda usando Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
```



```
        throw new RuntimeException(e);
    }
}
}
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

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 SQS com o Lambda usando JavaScript.

```
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.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }
}
```

```
private void processMessage(SQSMessage msg, Context context) {
    try {
        context.getLogger().log("Processed message " + msg.getBody());

        // TODO: Do interesting work based on the new message

    } catch (Exception e) {
        context.getLogger().log("An error occurred");
        throw e;
    }
}
```

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 fluxo 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 do Lambda que recebem eventos de uma fila do 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 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 [AWS Code Examples Repository](#).

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
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 = ""

                Usage:
                <mcRoleARN> <fileInput>\s

                Where:
                mcRoleARN - The MediaConvert Role ARN.\s
                fileInput - The URL of an Amazon S3 bucket
                where the input file is located.\s
                """;

        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)
            .build();

        String id = createMediaJob(mc, mcRoleARN, fileInput);
        System.out.println("MediaConvert job created. Job Id = " + id);
        mc.close();
    }

    public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
String fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
1) + "javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
        String mp4Output = s3path + "mp4/";

        try {
            DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service
endpoint URL!");

                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
            System.out.println("MediaConvert service URL: " +
endpointURL);

            System.out.println("MediaConvert role arn: " + mcRoleARN);
            System.out.println("MediaConvert input file: " + fileInput);
            System.out.println("MediaConvert output path: " + s3path);

            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            // output group Preset HLS low profile
            Output hlsLow = createOutput("hls_low", "_low", "$dt$",
750000, 7, 1920, 1080, 640);
            // output group Preset HLS media profile

```

```
        Output hlsMedium = createOutput("hls_medium", "_medium", "_
    $dt$", 1200000, 7, 1920, 1080, 1280);
        // output group Preset HLS high profole
        Output hlsHigh = createOutput("hls_high", "_high", "$dt$",
    3500000, 8, 1920, 1080, 1920);

        OutputGroup appleHLS = OutputGroup.builder().name("Apple
    HLS").customName("Example")

        .outputGroupSettings(OutputGroupSettings.builder()

        .type(OutputGroupType.HLS_GROUP_SETTINGS)

        .hlsGroupSettings(HlsGroupSettings.builder()

        .directoryStructure(

            HlsDirectoryStructure.SINGLE_DIRECTORY)

        .manifestDurationFormat(

            HlsManifestDurationFormat.INTEGER)

        .streamInfResolution(

            HlsStreamInfResolution.INCLUDE)

        .clientCache(HlsClientCache.ENABLED)

        .captionLanguageSetting(

            HlsCaptionLanguageSetting.OMIT)

        .manifestCompression(

            HlsManifestCompression.NONE)

        .codecSpecification(

            HlsCodecSpecification.RFC_4281)

        .outputSelection(

            HlsOutputSelection.MANIFESTS_AND_SEGMENTS)
```

```
.programDateTime(HlsProgramDateTime.EXCLUDE)

.programDateTimePeriod(600)

.timedMetadataId3Frame(
    HlsTimedMetadataId3Frame.PRIV)

.timedMetadataId3Period(10)

.destination(fileOutput)

.segmentControl(HlsSegmentControl.SEGMENTED_FILES)

.minFinalSegmentLength((double) 0)

.segmentLength(4).minSegmentLength(0).build()
                                                                    .build()
                                                                    .outputs(hlsLow, hlsMedium,
hlsHigh).build();

        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build()
                                                                    .build()
                                                                    .outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.MP4).build()

.videoDescription(VideoDescription.builder().width(1280)
                                                                    .height(720)

.scalingBehavior(ScalingBehavior.DEFAULT)
```

```
.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.H_264)

    .h264Settings(H264Settings

        .builder()

        .rateControlMode(
            H264RateControlMode.QVBR)

        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

        .qualityTuningLevel(
            H264QualityTuningLevel.SINGLE_PASS)

        .qvbrSettings(
            H264QvbrSettings.builder()

                .qvbrQualityLevel(
                    8)

                .build())

        .codecLevel(H264CodecLevel.AUTO)
```

```
.codecProfile(H264CodecProfile.MAIN)

.maxBitrate(2400000)

.framerateControl(
    H264FramerateControl.INITIALIZE_FROM_SOURCE)

.gopSize(2.0)

.gopSizeUnits(H264GopSizeUnits.SECONDS)

.numberBFramesBetweenReferenceFrames(
    2)

.gopClosedCadence(
    1)

.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED)

.syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(
    3)

.dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(
    H264SceneChangeDetect.ENABLED)

.minIInterval(0)

.telecine(H264Telecine.NONE)

.framerateConversionAlgorithm(
```

```
                H264FramerateConversionAlgorithm.DUPLICATE_DROP)

        .entropyEncoding(

                H264EntropyEncoding.CABAC)

        .slices(1)

        .unregisteredSeiTimecode(

                H264UnregisteredSeiTimecode.DISABLED)

        .repeatPps(H264RepeatPps.DISABLED)

        .adaptiveQuantization(

                H264AdaptiveQuantization.HIGH)

        .spatialAdaptiveQuantization(

                H264SpatialAdaptiveQuantization.ENABLED)

        .temporalAdaptiveQuantization(

                H264TemporalAdaptiveQuantization.ENABLED)

        .flickerAdaptiveQuantization(

                H264FlickerAdaptiveQuantization.DISABLED)

        .softness(0)

        .interlaceMode(H264InterlaceMode.PROGRESSIVE)

        .build())

        .build()

                .build()

        .audioDescriptions(AudioDescription.builder())

        .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
```

```

.languageCodeControl(
    AudioLanguageCodeControl.FOLLOW_INPUT)
.codecSettings(AudioCodecSettings.builder()
    .codec(AudioCodec.AAC)
    .aacSettings(AacSettings
        .builder()
        .codecProfile(AacCodecProfile.LC)
        .rateControlMode(
            AacRateControlMode.CBR)
        .codingMode(AacCodingMode.CODING_MODE_2_0)
        .sampleRate(44100)
        .bitrate(160000)
        .rawFormat(AacRawFormat.NONE)
        .specification(AacSpecification.MPEG4)
        .audioDescriptionBroadcasterMix(
            AacAudioDescriptionBroadcasterMix.NORMAL)
        .build())
    .build())
    .build())
    .build())
    .build();
    OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")
    .outputGroupSettings(OutputGroupSettings.builder()

```

```
.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())
                                .build())
                                .outputs(Output.builder().extension("jpg"))

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.FRAME_CAPTURE)

    .frameCaptureSettings(

        FrameCaptureSettings

            .builder()

                .framerateNumerator(

                    1)

                .framerateDenominator(

                    1)
```



```

        .maxCaptures(10000000)

        .quality(80)

        .build()

    .build()

        .build()

        .build();

    Map<String, AudioSelector> audioSelectors = new HashMap<>();
    audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
        .offset(0).build());

        JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
        .audioSelectors(audioSelectors)
        .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

        .rotate(InputRotate.DEGREE_0).build())

        .filterEnable(InputFilterEnable.AUTO).filterStrength(0)
        .deblockFilter(InputDeblockFilter.DISABLED)

        .denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

        .timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
        .outputGroups(appleHLS, thumbs,
fileMp4).build();

        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
        .settings(jobSettings)
        .build();

        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
        return createJobResponse.job().id();

```

```

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
        return "";
    }

    private final static Output createOutput(String customName,
        String nameModifier,
        String segmentModifier,
        int qvbrMaxBitrate,
        int qvbrQualityLevel,
        int originWidth,
        int originHeight,
        int targetWidth) {

        int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
        Output output = null;
        try {
            output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                                .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

```

```

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,
           489, 490, 491, 492)

                                           .build()
                                           .build()
                                           .videoDescription(
VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.H_264)

    .h264Settings(H264Settings

        .builder()

            .rateControlMode(

```

```
                H264RateControlMode.QVBR)

        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

        .qualityTuningLevel(

                H264QualityTuningLevel.SINGLE_PASS)

        .qvbrSettings(H264QvbrSettings

                .builder()

                .qvbrQualityLevel(

                        qvbrQualityLevel)

                .build())

        .codecLevel(H264CodecLevel.AUTO)

        .codecProfile((targetHeight > 720

                && targetWidth > 1280)

                ? H264CodecProfile.HIGH

                : H264CodecProfile.MAIN)

        .maxBitrate(qvbrMaxBitrate)

        .framerateControl(

                H264FramerateControl.INITIALIZE_FROM_SOURCE)

        .gopSize(2.0)

        .gopSizeUnits(H264GopSizeUnits.SECONDS)

        .numberBFramesBetweenReferenceFrames(

                2)

        .gopClosedCadence(
```

```
1)

.gopBReference(H264GopBReference.DISABLED)

.slowPal(H264SlowPal.DISABLED)

.syntax(H264Syntax.DEFAULT)

.numberReferenceFrames(

3)

.dynamicSubGop(H264DynamicSubGop.STATIC)

.fieldEncoding(H264FieldEncoding.PAFF)

.sceneChangeDetect(

    H264SceneChangeDetect.ENABLED)

.minIInterval(0)

.telecine(H264Telecine.NONE)

.framerateConversionAlgorithm(

    H264FramerateConversionAlgorithm.DUPLICATE_DROP)

.entropyEncoding(

    H264EntropyEncoding.CABAC)

.slices(1)

.unregisteredSeiTimecode(

    H264UnregisteredSeiTimecode.DISABLED)

.repeatPps(H264RepeatPps.DISABLED)

.adaptiveQuantization(

    H264AdaptiveQuantization.HIGH)
```

```
        .spatialAdaptiveQuantization(
            H264SpatialAdaptiveQuantization.ENABLED)
        .temporalAdaptiveQuantization(
            H264TemporalAdaptiveQuantization.ENABLED)
        .flickerAdaptiveQuantization(
            H264FlickerAdaptiveQuantization.DISABLED)
        .softness(0)
        .interlaceMode(H264InterlaceMode.PROGRESSIVE)
        .build()

    .build()
                                                                    .build()

    .audioDescriptions(AudioDescription.builder()
    .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
    .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)
    .codecSettings(AudioCodecSettings.builder()
    .codec(AudioCodec.AAC).aacSettings(AacSettings
        .builder()
        .codecProfile(AacCodecProfile.LC)
        .rateControlMode(
            AacRateControlMode.CBR)
        .codingMode(AacCodingMode.CODING_MODE_2_0)
        .sampleRate(44100)
```

```

        .bitrate(96000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(

            AacAudioDescriptionBroadcasterMix.NORMAL)

        .build())

        .build())

        .build())

        .build();
    } catch (MediaConvertException e) {
        e.printStackTrace();
        System.exit(0);
    }
    return output;
}
}

```

- Para obter detalhes da API, consulte [CreateJob](#) Referência AWS SDK for Java 2.x da API.

Obter 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 [AWS Code Examples Repository](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;

```

```
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * 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 GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
            " <jobId> \n\n" +
            "Where:\n" +
            " jobId - The job id value.\n\n";

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

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * 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 ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        listCompleteJobs(mc);
        mc.close();
    }

    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](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Migration Hub usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 stream de progresso

O exemplo de código a seguir mostra como excluir um stream de progresso.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * 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 DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <progressStream>\s

                Where:
                    progressStream - the name of a progress stream to delete.\s
                """;
```

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

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    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](#) Referência AWS SDK for Java 2.x da API.

Descrever 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * 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 DescribeAppState {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                DescribeAppState <appId>\s

                Where:
                appId - the application id value.\s
                """;

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

        String appId = args[0];
        Region region = Region.US_WEST_2;
```

```
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

describeApplicationState(migrationClient, appId);
migrationClient.close();
}

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](#) a Referência AWS SDK for Java 2.x da API.

Obter 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
    software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * 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 DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                DescribeMigrationTask <migrationTask> <progressStream>\s

            Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s

            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```



```

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Listar aplicações

O exemplo de código a seguir mostra como listar aplicações.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * 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 ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listApps(migrationClient);
        migrationClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
```

```
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup 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 ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    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 tarefas de migração.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * 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 ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
            .region(region)
            .build();
```

```
        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
    software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
    software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * 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 ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <migrationTask> <progressStream>\s

                Where:
                migrationTask - the name of a migration task.\s
                progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
```

```
String progressStream = args[1];
Region region = Region.US_WEST_2;
MigrationHubClient migrationClient = MigrationHubClient.builder()
    .region(region)
    .build();

importMigrTask(migrationClient, migrationTask, progressStream);
migrationClient.close();
}

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 [ImportMigrationTask](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Personalize usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 trabalho 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 [AWS Code Examples Repository](#).

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                String solutionVersionArn,
                String jobName,
                String s3InputDataSourcePath,
                String s3DataDestinationPath,
                String roleArn,
```

```
        String explorationWeight,
        String explorationItemAgeCutOff) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {

        // Set up data input and output parameters.
        S3DataConfig inputSource = S3DataConfig.builder()
            .path(s3InputDataSourcePath)
            .build();

        S3DataConfig outputDestination = S3DataConfig.builder()
            .path(s3DataDestinationPath)
            .build();

        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
            .s3DataSource(inputSource)
            .build();

        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
            .s3DataDestination(outputDestination)
            .build();

        // Optional code to build the User-Personalization specific
item exploration
        // config.
        HashMap<String, String> explorationConfig = new HashMap<>();

        explorationConfig.put("explorationWeight",
explorationWeight);
        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
            .itemExplorationConfig(explorationConfig)
            .build();

        // End optional User-Personalization recipe specific code.
```

```

        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
        .builder()
        .solutionVersionArn(solutionVersionArn)
        .jobInput(jobInput)
        .jobOutput(jobOutputLocation)
        .jobName(jobName)
        .roleArn(roleArn)
        .batchInferenceJobConfig(jobConfig) //
Optional
        .build();

        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
        .batchInferenceJobArn();

        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
        .builder()
        .batchInferenceJobArn(batchInferenceJobArn)
        .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
        while (Instant.now().getEpochSecond() < maxTime) {

                BatchInferenceJob batchInferenceJob =
personalizeClient
        .describeBatchInferenceJob(describeBatchInferenceJobRequest)
                .batchInferenceJob();

                status = batchInferenceJob.status();
                System.out.println("Batch inference job status: " +
status);

                if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {
                        break;
                }
                try {
                        Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                }
        }

```

```
        }
    }
    return batchInferenceJobArn;
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- 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 do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void createPersonalCampaign(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 [CreateCampaigna](#) Referência AWS SDK for Java 2.x da API.

Criar um conjunto de dados

O exemplo de código a seguir mostra como criar um 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 [AWS Code Examples Repository](#).

```
public static String createDataset(PersonalizeClient personalizeClient,  
    String datasetName,  
    String datasetGroupArn,  
    String datasetType,  
    String schemaArn) {  
    try {  
        CreateDatasetRequest request = CreateDatasetRequest.builder()  
            .name(datasetName)  
            .datasetGroupArn(datasetGroupArn)  
            .datasetType(datasetType)  
            .schemaArn(schemaArn)  
            .build();  
  
        String datasetArn = personalizeClient.createDataset(request)  
            .datasetArn();  
        System.out.println("Dataset " + datasetName + " created.");  
        return datasetArn;  
    } catch (PersonalizeException e) {
```

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

- Para obter detalhes da API, consulte [CreateDataset](#) 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 [AWS Code Examples Repository](#).

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
    String jobName,
    String datasetArn,
    IngestionMode ingestionMode,
    String roleArn,
    String s3BucketPath,
    String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {

        S3DataConfig exportS3DataConfig =
        S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
        DatasetExportJobOutput jobOutput =
        DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
            .build();
```

```
        CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
    .jobName(jobName)
    .datasetArn(datasetArn)
    .ingestionMode(ingestionMode)
    .jobOutput(jobOutput)
    .roleArn(roleArn)
    .build();

    String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

    DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
    .datasetExportJobArn(datasetExportJobArn)
    .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetExportJob datasetExportJob = personalizeClient
            .describeDatasetExportJob(describeDatasetExportJobRequest)
            .datasetExportJob();

        status = datasetExportJob.status();
        System.out.println("Export job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            return status;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- Para obter detalhes da API, consulte [CreateDatasetExportJob](#) Referência AWS SDK for Java 2.x da API.

Criar 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 [AWS Code Examples Repository](#).

```
public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Criar um grupo de conjunto 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](#) 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 [AWS Code Examples Repository](#).

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
    String jobName,
    String datasetArn,
    String s3BucketPath,
    String roleArn) {

    long waitInMilliseconds = 60 * 1000;
    String status;
```

```
String datasetImportJobArn;

try {
    DataSource importDataSource = DataSource.builder()
        .dataLocation(s3BucketPath)
        .build();

    CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
    .datasetArn(datasetArn)
    .dataSource(importDataSource)
    .jobName(jobName)
    .roleArn(roleArn)
    .build();

    datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
    .datasetImportJobArn();
    DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
    .datasetImportJobArn(datasetImportJobArn)
    .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetImportJob datasetImportJob = personalizeClient
            .describeDatasetImportJob(describeDatasetImportJobRequest)
            .datasetImportJob();

        status = datasetImportJob.status();
        System.out.println("Dataset import job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return datasetImportJobArn;
}
```

```
    } catch (PersonalizeException e) {  
        System.out.println(e.awsErrorDetails().errorMessage());  
    }  
    return "";  
}
```

- Para obter detalhes da API, consulte [CreateDatasetImportJob](#) 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 do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createDomainSchema(PersonalizeClient personalizeClient,  
String schemaName, String domain,  
String filePath) {  
  
    String schema = null;  
    try {  
        schema = new String(Files.readAllBytes(Paths.get(filePath)));  
    } catch (IOException e) {  
        System.out.println(e.getMessage());  
    }  
  
    try {  
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()  
            .name(schemaName)  
            .domain(domain)  
            .schema(schema)  
            .build();
```

```
        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

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

- Para obter detalhes da API, consulte [CreateSchema](#) a Referência AWS SDK for Java 2.x da API.

Criar um filtro

O exemplo de código a seguir mostra como criar um filtro do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createFilter(PersonalizeClient personalizeClient,
    String filterName,
    String datasetGroupArn,
    String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();
```

```

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

- Para obter detalhes da API, consulte [CreateFilter](#) Referência AWS SDK for Java 2.x da API.

Criar 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 [AWS Code Examples Repository](#).

```

public static String createRecommender(PersonalizeClient personalizeClient,
    String name,
    String datasetGroupArn,
    String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .name(name)
            .recipeArn(recipeArn)
            .build();

        CreateRecommenderResponse recommenderResponse = personalizeClient
            .createRecommender(createRecommenderRequest);
    }
}

```

```
String recommenderArn = recommenderResponse.recommenderArn();
System.out.println("The recommender ARN is " + recommenderArn);

DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
    .recommenderArn(recommenderArn)
    .build();

maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

while (Instant.now().getEpochSecond() < maxTime) {

    recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
    .status();
    System.out.println("Recommender status: " + recommenderStatus);

    if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return recommenderArn;

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

- Para obter detalhes da API, consulte [CreateRecommender](#) a Referência AWS SDK for Java 2.x da API.

Criar um esquema

O exemplo de código a seguir mostra como criar um esquema do Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

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

- Para obter detalhes da API, consulte [CreateSchema](#) a Referência AWS SDK for Java 2.x da API.

Criar uma solução

O exemplo de código a seguir mostra como criar uma solução do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
        String datasetGroupArn,
        String solutionName,
        String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

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

- Para obter detalhes da API, consulte [CreateSolution](#) na Referência AWS SDK for Java 2.x da API.

Criar uma versão da solução

O exemplo de código a seguir mostra como criar uma solução do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

```
        }
    }

    if (solutionStatus.equals("ACTIVE")) {

        CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
            .createSolutionVersion(createSolutionVersionRequest);
        solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

        System.out.println("Solution version ARN: " + solutionVersionArn);

        DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
            .solutionVersionArn(solutionVersionArn)
            .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                .solutionVersion().status();
            System.out.println("Solution version status: " +
solutionVersionStatus);

            if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return solutionVersionArn;
    }
} catch (PersonalizeException e) {
```

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

- Para obter detalhes da API, consulte [CreateSolutionVersion](#) na Referência AWS SDK for Java 2.x da API.

Criar um rastreador de eventos

O exemplo de código a seguir mostra como criar um rastreador de eventos do Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
    String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds
    String status;

    try {

        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
            .name(eventTrackerName)
            .datasetGroupArn(datasetGroupArn)
            .build();

        CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
```

```
        .createEventTracker(createEventTrackerRequest);

    eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
    eventTrackerId = createEventTrackerResponse.trackingId();
    System.out.println("Event tracker ARN: " + eventTrackerArn);
    System.out.println("Event tracker ID: " + eventTrackerId);

    maxTime = Instant.now().getEpochSecond() + maxTime;

    DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
        .eventTrackerArn(eventTrackerArn)
        .build();

    while (Instant.now().getEpochSecond() < maxTime) {

        status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
        System.out.println("EventTracker status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return eventTrackerId;
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return eventTrackerId;
}
```

- Para obter detalhes da API, consulte [CreateEventTracker](#) na 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 [AWS Code Examples Repository](#).

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        personalizeClient.deleteCampaign(campaignRequest);

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

- Para obter detalhes da API, consulte [DeleteCampaigna](#) 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 [AWS Code Examples Repository](#).

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        personalizeClient.deleteSolution(solutionRequest);
        System.out.println("Done");


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

- 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 [AWS Code Examples Repository](#).

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);

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

- Para obter detalhes da API, consulte [DeleteEventTracker](#) a Referência AWS SDK for Java 2.x da API.

Descrever 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 [AWS Code Examples Repository](#).

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
```

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

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is " + myCampaign.name());
        System.out.println("The Campaign status is " + myCampaign.status());

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

- Para obter detalhes da API, consulte [DescribeCampaign](#) Referência AWS SDK for Java 2.x da API.

Descrever uma fórmula

O exemplo de código a seguir mostra como descrever uma fórmula no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

    try {
        DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
            .recipeArn(recipeArn)
            .build();

        DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
```



```
        System.out.println("The recipe name is " +
recipeResponse.recipe().name());

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

- Para obter detalhes da API, consulte [DescribeRecipe](#) a Referência AWS SDK for Java 2.x da API.

Descreve 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 [AWS Code Examples Repository](#).

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

    try {
        DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
        System.out.println("The Solution name is " +
response.solution().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

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

- 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 [AWS Code Examples Repository](#).

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

    try {
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
            .solutionArn(solutionArn)
            .build();

        ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
        List<CampaignSummary> campaigns = response.campaigns();
        for (CampaignSummary campaign : campaigns) {
            System.out.println("Campaign name is : " + campaign.name());
            System.out.println("Campaign ARN is : " + campaign.campaignArn());
        }

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

```
}
```

- Para obter detalhes da API, consulte [ListCampaigns](#) na Referência AWS SDK for Java 2.x da API.

Listar grupos de conjunto 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 [AWS Code Examples Repository](#).

```
public static void listDSGroups(PersonalizeClient personalizeClient) {  
  
    try {  
        ListDatasetGroupsRequest groupsRequest =  
ListDatasetGroupsRequest.builder()  
            .maxResults(15)  
            .build();  
  
        ListDatasetGroupsResponse groupsResponse =  
personalizeClient.listDatasetGroups(groupsRequest);  
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();  
        for (DatasetGroupSummary group : groups) {  
            System.out.println("The DataSet name is : " + group.name());  
            System.out.println("The DataSet ARN is : " +  
group.datasetGroupArn());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [ListDatasetGroups](#) na Referência AWS SDK for Java 2.x da API.

Listar receitas

O exemplo de código a seguir mostra como listar fórmulas no Amazon Personalize.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {  
  
    try {  
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()  
            .maxResults(15)  
            .build();  
  
        ListRecipesResponse response =  
personalizeClient.listRecipes(recipesRequest);  
        List<RecipeSummary> recipes = response.recipes();  
        for (RecipeSummary recipe : recipes) {  
            System.out.println("The recipe ARN is: " + recipe.recipeArn());  
            System.out.println("The recipe name is: " + recipe.name());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [ListRecipes](#) na 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 [AWS Code Examples Repository](#).

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

    try {
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
            .maxResults(10)
            .datasetGroupArn(datasetGroupArn)
            .build();

        ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
        List<SolutionSummary> solutions = response.solutions();
        for (SolutionSummary solution : solutions) {
            System.out.println("The solution ARN is: " +
solution.solutionArn());
            System.out.println("The solution name is: " + solution.name());
        }

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

- Para obter detalhes da API, consulte [ListSolutions](#) na Referência AWS SDK for Java 2.x da API.

Atualizar uma campanha

O exemplo de código a seguir mostra como atualizar uma campanha do Amazon Personalize.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String updateCampaign(PersonalizeClient personalizeClient,
    String campaignArn,
    String solutionVersionArn,
    Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign updatedCampaign = campaignResponse.campaign();

        System.out.println("The Campaign status is " +
updatedCampaign.status());
        return updatedCampaign.status();

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

```
        return "";  
    }
```

- Para obter detalhes da API, consulte [UpdateCampaign](#) na 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Importar 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 [AWS Code Examples Repository](#).

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String item1Id,
    String item1PropertyName,
    String item1PropertyValue,
    String item2Id,
    String item2PropertyName,
    String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{\ \"%1$s\": \"%2$s
\}"),
                item1PropertyName,
                item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{\ \"%1$s\": \"%2$s
\}"),
                item2PropertyName,
                item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
            .datasetArn(datasetArn)
            .items(items)
            .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    }
}
```



```

        } catch (PersonalizeEventsException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return responseCode;
    }
}

```

- Para obter detalhes da API, consulte [PutEvents](#) na 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 eventos do Amazon Personalize Events.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String user1Id,
    String user1PropertyName,
    String user1PropertyValue,
    String user2Id,
    String user2PropertyName,
    String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
            .userId(user1Id)
            .properties(String.format("{ \"%1$s\": \"%2$s
\}"}",
                user1PropertyName,
                user1PropertyValue))

```

```

        .build());

        users.add(user1);

        User user2 = User.builder()
            .userId(user2Id)
            .properties(String.format("{\\\"%1$s\\\": \\\"%2$s
\\}\",
                                user2PropertyName,
                                user2PropertyValue))
            .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
            .datasetArn(datasetArn)
            .users(users)
            .build();

        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}

```

- Para obter detalhes da API, consulte [PutUsers](#) na Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Personalize Runtime usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Obter recomendações (grupo de conjunto de dados personalizados)

O exemplo de código a seguir mostra como obter recomendações de classificação do Amazon Personalize Runtime.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
        String campaignArn,
        String userId,
        ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
        GetPersonalizedRankingRequest.builder()
            .campaignArn(campaignArn)
            .userId(userId)
            .inputList(items)
            .build();

        GetPersonalizedRankingResponse recommendationsResponse =
        personalizeRuntimeClient
            .getPersonalizedRanking(rankingRecommendationsRequest);
```

```

        List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
            System.out.println("-----");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

```

- Para obter detalhes da API, consulte [GetPersonalizedRanking](#) Referência AWS SDK for Java 2.x da API.

Receber recomendações de um recomendador (grupo de conjunto de dados de domínio)

O exemplo de código a seguir mostra como obter recomendações de runtime do Amazon Personalize Runtime.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obter uma lista de 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);
    }
}

```

Obter 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);
    }
}

```

Usar 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 para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

Crie uma campanha.

```
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;

/**
 * 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 CreateCampaign {
    public static void main(String[] args) {

        final String usage = ""

            Usage:  <appId> <segmentId>
```



```
        Where:
            appId - The ID of the application to create the campaign in.
            segmentId - The ID of the segment to create the campaign from.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String appId = args[0];
    String segmentId = args[1];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createPinCampaign(pinpoint, appId, segmentId);
    pinpoint.close();
}

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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreateSegment {
    public static void main(String[] args) {
        final String usage = ""

                Usage:  <appId>

                Where:
                    appId - The application ID to create a segment
for.

                """;

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

        String appId = args[0];
```

```
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        SegmentResponse result = createSegment(pinpoint, appId);
        System.out.println("Segment " + result.name() + " created.");
        System.out.println(result.segmentType());
        pinpoint.close();
    }

    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 [CreateSegmenta](#) Referência AWS SDK for Java 2.x da API.

Cria uma aplicação

O exemplo de código a seguir mostra como criar uma aplicação.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreateApp {
    public static void main(String[] args) {
        final String usage = ""

                Usage: <appName>

                Where:
                appName - The name of the application to create.

                "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    String appID = createApplication(pinpoint, appName);
    System.out.println("App ID is: " + appID);
    pinpoint.close();
}

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.

Deleta o aplicativo

O exemplo de código a seguir mostra como excluir uma aplicação.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Excluir uma aplicação.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * 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 DeleteApp {
    public static void main(String[] args) {
        final String usage = ""

            Usage: <appId>

            Where:
            appId - The ID of the application to delete.

            "";

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

        String appId = args[0];
        System.out.println("Deleting an application with ID: " + appId);
        PinpointClient pinpoint = PinpointClient.builder()
```



```
        .region(Region.US_EAST_1)
        .build();

deletePinApp(pinpoint, appId);
System.out.println("Done");
pinpoint.close();
}

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 [AWS Code Examples Repository](#).

Excluir um endpoint

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * 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 DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appName> <endpointId >

            Where:
                appId - The id of the application to delete.
                endpointId - The id of the endpoint to delete.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deletePinEndpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    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

Os exemplos a seguir mostram 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 [AWS Code Examples Repository](#).

Exportar um endpoint.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
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.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-export.html
 *
 * 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 ExportEndpoints {
    public static void main(String[] args) {
        final String usage = ""

        This program performs the following steps:

        1. Exports the endpoints to an Amazon S3 bucket.
        2. Downloads the exported endpoints files from Amazon S3.
    }
}
```

3. Parses the endpoints files to obtain the endpoint IDs and prints them.

```
Usage: ExportEndpoints <applicationId> <s3BucketName>
<iamExportRoleArn> <path>
```

Where:

applicationId - The ID of the Amazon Pinpoint application that has the endpoint.

s3BucketName - The name of the Amazon S3 bucket to export the JSON file to.\s

iamExportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint write permissions to the S3 bucket. path - The path where the files downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).

```
""";
```

```
if (args.length != 4) {
    System.out.println(usage);
    System.exit(1);
}

String applicationId = args[0];
String s3BucketName = args[1];
String iamExportRoleArn = args[2];
String path = args[3];
System.out.println("Deleting an application with ID: " + applicationId);

Region region = Region.US_EAST_1;
PinpointClient pinpoint = PinpointClient.builder()
    .region(region)
    .build();

S3Client s3Client = S3Client.builder()
    .region(region)
    .build();

exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
pinpoint.close();
s3Client.close();
}

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.

Obter 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 [AWS Code Examples Repository](#).

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * 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 LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <endpoint>

            Where:
                appId - The ID of the application to delete.
                endpoint - The ID of the endpoint.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }

    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 [AWS Code Examples Repository](#).

Importe um segmento.

```
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;
```

```
/**
 * 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 ImportSegment {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId> <bucket> <key> <roleArn>\s

            Where:
                appId - The application ID to create a segment for.
                bucket - The name of the Amazon S3 bucket that contains the
segment definitons.
                key - The key of the S3 object.
                roleArn - ARN of the role that allows Amazon Pinpoint to
access S3. You need to set trust management for this to work. See https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_policies\_elements\_principal.html
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
import job status.");
    }
}
```

```
        System.out.println("See application " + response.jobStatus() + " for import
job status.");
        pinpoint.close();
    }

    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](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * 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 ListEndpointIds {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <applicationId> <userId>

                Where:
                    applicationId - The ID of the Amazon Pinpoint application that
has the endpoint.
                    userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String applicationId = args[0];
String userId = args[1];
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

listAllEndpoints(pinpoint, applicationId, userId);
pinpoint.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

Listar segmentos.

```
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;

/**
 * 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 ListSegments {
    public static void main(String[] args) {
        final String usage = ""

            Usage:  <appId>

            Where:
                appId - The ID of the application that contains a segment.

            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String appId = args[0];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    listSegs(pinpoint, appId);
    pinpoint.close();
}

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](#) na Referência AWS SDK for Java 2.x da API.

Enviar e-mails e mensagens de texto

O exemplo de código a seguir mostra como enviar e-mail e mensagens de texto 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 [AWS Code Examples Repository](#).

Envie uma mensagem de e-mail.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * 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 SendEmailMessage {

    // The character encoding the you want to use for the subject line and
```

```

// message body of the email.
public static String charset = "UTF-8";

// The body of the email for recipients whose email clients support HTML
content.
static final String body = ""
    Amazon Pinpoint test (AWS SDK for Java 2.x)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK
for Java 2.x

    """;

public static void main(String[] args) {
    final String usage = ""

        Usage:    <subject> <appId> <senderAddress>
<toAddress>

        Where:
            subject - The email subject to use.
            senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
            toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
        """;

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

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    sendEmail(pinpoint, subject, senderAddress, toAddress);
    System.out.println("Email was sent");
    pinpoint.close();
}

```

```
public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
            .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

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

Envie uma mensagem de e-mail com cópia para outros destinatários.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * 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 SendEmailMessageCC {

    // The body of the email.
    static final String body = ""
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
    for Java 2.x

    """;
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
                subject - The email subject to use.
                senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
                toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
                ccAddress - The CC address.

            """;
```

```
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String subject = args[0];
    String senderAddress = args[1];
    String toAddress = args[2];
    String ccAddress = args[3];

    System.out.println("Sending a message");
    PinpointEmailClient pinpoint = PinpointEmailClient.builder()
        .region(Region.US_EAST_1)
        .build();

    ArrayList<String> ccList = new ArrayList<>();
    ccList.add(ccAddress);
    sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
    pinpoint.close();
}

public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
    try {
        Content content = Content.builder()
            .data(body)
            .build();

        Body messageBody = Body.builder()
            .text(content)
            .build();

        Message message = Message.builder()
            .body(messageBody)
            .subject(Content.builder().data(subject).build())
            .build();

        Destination destination = Destination.builder()
            .toAddresses(toAddress)
            .ccAddresses(ccAddresses)
            .build();

        EmailContent emailContent = EmailContent.builder()
            .simple(message)
```

```
        .build();

        SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(senderAddress)
            .destination(destination)
            .content(emailContent)
            .build();

        pinpointEmailClient.sendEmail(sendEmailRequest);
        System.out.println("Message Sent");

    } catch (PinpointException e) {
        // Handle exception
        e.printStackTrace();
    }
}
}
```

Envie uma mensagem SMS.

```
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.SendMessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessageResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * 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 SendMessage {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
    countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber>\s

            Where:
                message - The body of the message to send.
                appId - The Amazon Pinpoint project/application ID
to use when you send this message.
                originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
                destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];
        String originationNumber = args[2];
        String destinationNumber = args[3];
        System.out.println("Sending a message");
    }
}

```



```
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
        pinpoint.close();
    }

    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.sendMessage(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);
    }
}

```

Enviar mensagens SMS em lote.

```

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;

/**
 * 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 SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
    countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber> <destinationNumber1>\s

            Where:
            message - The body of the message to send.
            appId - The Amazon Pinpoint project/application ID
to use when you send this message.
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
            destinationNumber1 - The second recipient's phone
number. For best results, you should specify the phone number in E.164 format (for
example, +1-555-555-5654).\s

            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];

```

```

        String originationNumber = args[2];
        String destinationNumber = args[3];
        String destinationNumber1 = args[4];
        System.out.println("Sending a message");
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
        pinpoint.close();
    }

    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.sendMessage(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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * 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 UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = ""

                Usage: <appId>

                Where:
                    appId - The ID of the application to create an endpoint for.

                """;

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

```
    }

    String appId = args[0];
    PinpointClient pinpoint = PinpointClient.builder()
        .region(Region.US_EAST_1)
        .build();

    EndpointResponse response = createEndpoint(pinpoint, appId);
    System.out.println("Got Endpoint: " + response.id());
    pinpoint.close();
}

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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * 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 UpdateChannel {
    public static void main(String[] args) {
        final String usage = ""

                Usage: CreateChannel <appId>

                Where:
                appId - The name of the application whose channel is updated.

                """;

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

        String appId = args[0];
```

```
PinpointClient pinpoint = PinpointClient.builder()
    .region(Region.US_EAST_1)
    .build();

SMSChannelResponse getResponse = getSmsChannel(pinpoint, appId);
toggleSmsChannel(pinpoint, appId, getResponse);
pinpoint.close();
}

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](#) na Referência AWS SDK for Java 2.x da API.

Exemplos da API SMS and Voice do Amazon Pinpoint usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK for Java 2.x de voz e SMS do 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 uma mensagem de voz com Amazon Pinpoint SMS e Voice API

O exemplo de código a seguir mostra como enviar uma mensagem de voz com a API SMS and Voice do Amazon Pinpoint.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * 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 SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
    list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";
```

```

// The content of the message. This example uses SSML to customize and
control
// certain aspects of the message, such as by adding pauses and changing
// phonation. The message can't contain any line breaks.
static final String ssmlMessage = "<speak>This is a test message sent from "
    + "<emphasis>Amazon Pinpoint</emphasis> "
    + "using the <break strength='weak'>AWS "
    + "SDK for Java. "
    + "<amazon:effect phonation='soft'>Thank "
    + "you for listening.</amazon:effect></speak>";

public static void main(String[] args) {

    final String usage = ""

        Usage:  <originationNumber> <destinationNumber>\s

        Where:
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String originationNumber = args[0];
    String destinationNumber = args[1];
    System.out.println("Sending a voice message");

    // Set the content type to application/json.
    List<String> listVal = new ArrayList<>();
    listVal.add("application/json");
    Map<String, List<String>> values = new HashMap<>();
    values.put("Content-Type", listVal);

    ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
        .headers(values)

```

```
        .build();

        PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
            .overrideConfiguration(config2)
            .region(Region.US_EAST_1)
            .build();

        sendVoiceMsg(client, originationNumber, destinationNumber);
        client.close();
    }

    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 para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Obter vozes disponíveis para síntese

O exemplo de código a seguir mostra como obter vozes disponíveis para síntese do Amazon Polly.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
```



```
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * 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 DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        describeVoice(polly);
        polly.close();
    }

    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 [DescrevaVoices](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * 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 ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listLexicons(polly);
        polly.close();
    }

    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 [AWS Code Examples Repository](#).

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
```

```
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * 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 PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
        built this working demo " +
        " of Amazon Polly in Java Version 2. Have fun building voice enabled
        apps with Amazon Polly (that's me!), and always "
        +
        " look at the AWS website for tips and tricks on using Amazon Polly and
        other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
                .engine("standard")
                .build();
```

```

        DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
        Voice voice = describeVoicesResult.voices().stream()
            .filter(v -> v.name().equals("Joanna"))
            .findFirst()
            .orElseThrow(() -> new RuntimeException("Voice not found"));
        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](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon RDS usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;
```

```
/**
 * 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 DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    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 da API do AWS SDK for Java 2.x .

Tópicos

- [Ações](#)
- [Cenários](#)

Ações

Criar uma instância de banco de dados

O exemplo de código a seguir mostra como criar uma instância do Amazon RDS DB e aguardar até que se torne disponível.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * 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 more details, see:
*
* https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-
services-use-secrets\_RS.html
*
*
*/

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier> <dbName> <secretName>

            Where:
                dbInstanceIdentifier - The database instance identifier.\s
                dbName - The database name.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials."
            """;

        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)
            .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().dbInstanceStatus());

    } 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.dbInstanceStatus();
                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 da 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 do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());


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

- 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 do Amazon RDS DB.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

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

- Para ter detalhes da API, consulte [CreateDBSnapshot](#) na Referência da 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 [AWS Code Examples Repository](#).

Use a [RdsUtilities](#) classe para gerar um token de autenticação.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        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)
            .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 do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
```

```
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * 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 DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <dbInstanceIdentifier>\s

            Where:
                dbInstanceIdentifier - The database instance identifier\s
            """;

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

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
```



```

        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

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

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

```

- Para ter detalhes da API, consulte [DeleteDBInstance](#) na Referência da 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 Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```

// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
// exists.
public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
    throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN;

```

```

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            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 instâncias do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * 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 DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    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 da 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 um grupo de parâmetros do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
```

```
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

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

- 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 de 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 [AWS Code Examples Repository](#).

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

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
    }
}
```

```
List<DBEngineVersion> engines = response.dbEngineVersions();

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

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

- 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 opções para instâncias do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .dbParameterGroupFamily(dbParameterGroupFamily)
```

```
        .engine("mysql")
        .build();

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

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

- 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 os parâmetros em um grupo de parâmetros do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
```

```

        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

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

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

```

- Para ter detalhes da API, consulte [DescribeDBParameters](#) na Referência da 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 do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * 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 ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                dbInstanceIdentifier - The database instance identifier.\s
                masterUserPassword - The updated password that corresponds to
the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String dbInstanceIdentifier = args[0];
String masterUserPassword = args[1];
Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .build();

updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
rdsClient.close();
}

public static void updateIntance(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUserPassword) {
    try {
        // For a demo - modify the DB instance by modifying the master password.
        ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .publiclyAccessible(true)
        .masterUserPassword(masterUserPassword)
        .build();

        ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);
        System.out.print("The ARN of the modified database is: " +
instanceResponse.dbInstance().dbInstanceArn());

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

- Para ter detalhes da API, consulte [ModifyDBInstance](#) na Referência da API do AWS SDK for Java 2.x .

Reinicializar uma instância de banco de dados

Os exemplos de código a seguir mostram como reinicializar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * 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 RebootDBInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier>\s

                Where:
                dbInstanceIdentifier - The database instance identifier\s
                """;

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

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
```

```
        .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    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 da 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

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * 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 DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    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 os parâmetros em um grupo de parâmetros do Amazon RDS DB.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

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

```
}  
}
```

- 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 [AWS Code Examples Repository](#).

Execute várias operações.

```
import com.google.gson.Gson;  
import  
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.rds.RdsClient;  
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;  
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;  
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;  
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;  
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
```

```
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * 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 = ""

            Usage:
                <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier> <secretName>

            Where:
                dbGroupName - The database group name.\s
                dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
                dbInstanceIdentifier - The database instance identifier\s
                dbName - The database name.\s
                dbSnapshotIdentifier - The snapshot identifier.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
        """;

```

```
    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)
        .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.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

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

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

        DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
```

```
        List<DBSnapshot> snapshotList = response.dbSnapshots();
        for (DBSnapshot snapshot : snapshotList) {
            snapshotReadyStr = snapshot.status();
            if (snapshotReadyStr.contains("available")) {
                snapshotReady = true;
            } else {
                System.out.print(".");
                Thread.sleep(sleepTime * 1000);
            }
        }
    }

    System.out.println("The Snapshot is available!");
} catch (RdsException | InterruptedException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

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

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
```

```

        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();

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

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

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
        String dbGroupName,
        String dbInstanceIdentifier,
        String dbName,
        String masterUsername,
        String masterUserPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()

```



```

        .dbInstanceIdentifier(dbInstanceIdentifier)
        .allocatedStorage(100)
        .dbName(dbName)
        .dbParameterGroupName(dbGroupName)
        .engine("mysql")
        .dbInstanceClass("db.m4.large")
        .engineVersion("8.0")
        .storageType("standard")
        .masterUsername(masterUsername)
        .masterUserPassword(masterUserPassword)
        .build();

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

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

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
            .builder()
            .engine("mysql")
            .build();

        DescribeOrderableDbInstanceOptionsResponse response = rdsClient
            .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
        List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
        for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
        {
            System.out.println("The engine version is " +
dbInstanceOption.engineVersion());

```

```
        System.out.println("The engine description is " +
dbInstanceOption.engine());
    }

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

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

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

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

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();
```

```
        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

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

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
```

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

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

public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

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

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

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

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

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

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

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da 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 para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

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 [AWS Code Examples Repository](#).

Crie o cluster.

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
    String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername) // set the user name here
            .masterUserPassword(masterUserPassword) // set the user password
here
            .nodeType("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](#) a 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 [AWS Code Examples Repository](#).

Excluir o cluster.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;

/**
 * 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 DeleteCluster {
    public static void main(String[] args) {

        final String usage = ""

```



```
Usage:
    <clusterId>\s

Where:
    clusterId - The id of the cluster to delete.\s
""";

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

String clusterId = args[0];
Region region = Region.US_WEST_2;
RedshiftClient redshiftClient = RedshiftClient.builder()
    .region(region)
    .build();

deleteRedshiftCluster(redshiftClient, clusterId);
redshiftClient.close();
}

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](#) a Referência AWS SDK for Java 2.x da API.

Descrever seus clusters

O exemplo de código a seguir mostra como descrever 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 [AWS Code Examples Repository](#).

Descrever o cluster.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.Cluster;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import java.util.List;

/**
 * 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 DescribeClusters {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        describeRedshiftClusters(redshiftClient);
    }
}
```

```
        redshiftClient.close();
    }

    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](#) a 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 [AWS Code Examples Repository](#).

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](#) na 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Comparar 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 usando o Amazon Rekognition.

Para obter mais informações, consulte [Comparar faces em imagens](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 CompareFaces {
    public static void main(String[] args) {
        final String usage = ""
```

```

        Usage:    <pathSource> <pathTarget>

        Where:
            pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png).\\s
            pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png).\\s
        """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
        rekClient.close();
    }

    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](#) a 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 [Criar uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * 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 CreateCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionName>\s

                Where:
                    collectionName - The name of the collection.\s
                """;

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

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
```



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

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    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](#) na 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 mais informações, consulte [Excluir uma coleção](#).

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * 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 DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>\s

                Where:
                    collectionId - The id of the collection to delete.\s
                """;

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

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Excluir faces de uma coleção

O exemplo de código a seguir mostra como excluir faces de uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Excluir 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * 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 DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <faceId>\s

                Where:
                    collectionId - The id of the collection from which faces are
deleted.\s

                    faceId - The id of the face to delete.\s
                """;

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

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }
}
```

```
public static void deleteFacesCollection(RecognitionClient 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 (RecognitionException 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.

Descrever uma coleção

O exemplo de código a seguir mostra como descrever uma coleção do Amazon Rekognition.

Para obter mais informações, consulte [Descrever uma coleção](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
/**
 * 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 DescribeCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
                """;

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

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    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 faces em uma imagem com o Amazon Rekognition.

Para obter mais informações, consulte [Detectar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 DetectFaces {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
                \\pic1.png).\\s
                """;

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

    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.

Para 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 [Detectar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectImageLabels(rekClient, sourceImage);
    rekClient.close();
}

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](#) na 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 o conteúdo que pode ser inadequado para alguns públicos.

Para obter mais informações, consulte [Detectar imagens inapropriadas](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }

    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](#) na 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 [Detectar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 DetectText {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png).\s
                """;

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectTextLabels(rekClient, sourceImage);
        rekClient.close();
    }
}
```

```
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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

            Usage:      <collectionId> <sourceImage>

            Where:
                collectionName - The name of the collection.
```

```
        sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png)\\.\\s
        """";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    addToCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

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](#) na Referência AWS SDK for Java 2.x da API.

Listar coleções

O exemplo de código a seguir mostra como listar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * 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 ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Listar faces em uma coleção

O exemplo de código a seguir mostra como lista 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * 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 ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>

                Where:
```

```

        collectionId - The name of the collection.\s
        """;

    if (args.length < 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Faces in collection " + collectionId);
    listFacesCollection(rekClient, collectionId);
    rekClient.close();
}

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](#) Referência AWS SDK for Java 2.x da API.

Reconhecer 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;

/**
 * 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 RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <sourceImage>
```

```

        Where:
            sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s
            """;

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

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, sourceImage);
        rekClient.close();
    }

    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.

Pesquisar faces 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 [Pesquisar uma face \(Face ID\)](#).

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }
}
```

```

    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.

Pesquisar faces em uma coleção em comparação com uma imagem de referência

O exemplo de código a seguir mostra como pesquisar faces em uma coleção do Amazon Rekognition em comparação com uma imagem de referência.

Para obter mais informações, consulte [Pesquisar uma face \(imagem\)](#).

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * 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 SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
```

```
String faceId = args[1];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Searching for a face in a collections");
searchFaceById(rekClient, collectionId, faceId);
rekClient.close();
}

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](#) Referência AWS SDK for Java 2.x da API.

Cenários

Detectar 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.
- Verificar o status do trabalho até que eles terminem.
- Criar uma lista de elementos detectados por cada trabalho.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Obter resultados de celebridades a partir de um vídeo localizado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
    software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
    software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;
```

```
/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that 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 VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
    }
}
```

```
NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startCelebrityDetection(rekClient, channel, bucket, video);
getCelebrityDetectionResults(rekClient);
System.out.println("This example is done!");
rekClient.close();
}

public static void startCelebrityDetection(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();

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
            .notificationChannel(channel)
            .video(vidObj)
            .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);
    }
}
}

```

Detectar rótulos em um vídeo por meio de uma operação de detecção de rótulos.

```

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;

```

```
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * 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 VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
```

```
String topicArn = args[3];
String roleArn = args[4];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

SqsClient sqs = SqsClient.builder()
    .region(Region.US_EAST_1)
    .build();

NotificationChannel channel = NotificationChannel.builder()
    .snsTopicArn(topicArn)
    .roleArn(roleArn)
    .build();

startLabels(rekClient, channel, bucket, video);
getLabelJob(rekClient, sqs, queueUrl);
System.out.println("This example is done!");
sqs.close();
rekClient.close();
}

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(RecognitionClient 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());
            }
        } while (labelDetectionResult.nextToken() != null);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```

        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);
    }
}
}
}

```

Detectar faces em um vídeo armazenado em um bucket do Amazon S3.

```
import com.fasterxml.jackson.core.JsonProcessingException;
```



```
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
    software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * 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 VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

            Where:
```

```

        bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
        video - The name of the video (for example, people.mp4).\s
        queueUrl- The URL of a SQS queue.\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 5) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String queueUrl = args[2];
    String topicArn = args[3];
    String roleArn = args[4];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startLabels(rekClient, channel, bucket, video);
    getLabelJob(rekClient, sqs, queueUrl);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

public static void startLabels(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();

    StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
    .jobTag("DetectingLabels")
    .notificationChannel(channel)
    .video(vid0b)
    .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);
    }
}
}

```

Detectar conteúdo inapropriado ou ofensivo em um vídeo armazenado em um bucket do Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
    software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * 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 VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }
}
```



```
public static void startModerationDetection(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();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vid0b)
            .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);
        }
    }
}

```

Detectar segmentos de sinal técnico e segmentos de detecção de tomada em um vídeo armazenado em um bucket do Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
    software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
    software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * 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 VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:
                bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                video - The name of video (for example, people.mp4).\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        SqsClient sqs = SqsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();
    }
}
```

```
        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startSegmentDetection(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();

            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(vidObj)
                .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);
        }
    }
}

```

Detectar texto em um vídeo armazenado em um bucket do Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * 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 VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = ""

```



```

        Usage:    <bucket> <video> <topicArn> <roleArn>

        Where:
            bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
            video - The name of video (for example, people.mp4).\s
            topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
            roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        NotificationChannel channel = NotificationChannel.builder()
            .snsTopicArn(topicArn)
            .roleArn(roleArn)
            .build();

        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(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();

    StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .video(vid0b)
        .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 bucket do Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
    software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * 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 VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <bucket> <video> <topicArn> <roleArn>

            Where:

```

```

        bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
        video - The name of video (for example, people.mp4).\s
        topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
        roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
        """;

    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startPersonLabels(rekClient, channel, bucket, video);
    getPersonDetectionResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

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 da API do AWS SDK for Java 2.x .
 - [GetCelebrityRecognition](#)
 - [GetContentModeration](#)
 - [GetLabelDetection](#)
 - [GetPersonTracking](#)
 - [GetSegmentDetection](#)
 - [GetTextDetection](#)
 - [StartCelebrityRecognition](#)
 - [StartContentModeration](#)
 - [StartLabelDetection](#)
 - [StartPersonTracking](#)
 - [StartSegmentDetection](#)
 - [StartTextDetection](#)

Exemplos de registro de domínios do Route 53 usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * 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)
        .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](#) a 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 [AWS Code Examples Repository](#).

```
public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();
```

```
        CheckDomainAvailabilityResponse response = route53DomainsClient
            .checkDomainAvailability(availabilityRequest);
        System.out.println(domainSuggestion + " is " +
response.availability().toString());

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [CheckDomainAvailability](#) a Referência AWS SDK for Java 2.x da API.

Verificar a transferibilidade de um domínio

O exemplo de código a seguir mostra como verificar o potencial para transferência de um domínio.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
            .domainName(domainSuggestion)
            .build();

        CheckDomainTransferabilityResponse response = route53DomainsClient
            .checkDomainTransferability(transferabilityRequest);
        System.out.println("Transferability: " +
response.transferability().transferable().toString());
    }
```

```
    } catch (Route53Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- Para obter detalhes da API, consulte [CheckDomainTransferability](#) a 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 [AWS Code Examples Repository](#).

```
public static void getDomainDetails(Route53DomainsClient route53DomainsClient,  
String domainSuggestion) {  
    try {  
        GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()  
            .domainName(domainSuggestion)  
            .build();  
  
        GetDomainDetailResponse response =  
route53DomainsClient.getDomainDetail(detailRequest);  
        System.out.println("The contact first name is " +  
response.registrantContact().firstName());  
        System.out.println("The contact last name is " +  
response.registrantContact().lastName());  
        System.out.println("The contact org name is " +  
response.registrantContact().organizationName());  
  
    } catch (Route53Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

```
}  
}
```

- Para obter detalhes da API, consulte [GetDomainDetail](#) a Referência AWS SDK for Java 2.x da API.

Obter detalhes de uma operação

O exemplo de código a seguir mostra como obter os detalhes de uma operação.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void getOperationalDetail(Route53DomainsClient  
route53DomainsClient, String operationId) {  
    try {  
        GetOperationDetailRequest detailRequest =  
GetOperationDetailRequest.builder()  
            .operationId(operationId)  
            .build();  
  
        GetOperationDetailResponse response =  
route53DomainsClient.getOperationDetail(detailRequest);  
        System.out.println("Operation detail message is " + response.message());  
  
    } catch (Route53Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- 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 [AWS Code Examples Repository](#).

```
public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
    try {
        GetDomainSuggestionsRequest suggestionsRequest =
        GetDomainSuggestionsRequest.builder()
            .domainName(domainSuggestion)
            .suggestionCount(5)
            .onlyAvailable(true)
            .build();

        GetDomainSuggestionsResponse response =
        route53DomainsClient.getDomainSuggestions(suggestionsRequest);
        List<DomainSuggestion> suggestions = response.suggestionsList();
        for (DomainSuggestion suggestion : suggestions) {
            System.out.println("Suggestion Name: " + suggestion.domainName());
            System.out.println("Availability: " + suggestion.availability());
            System.out.println(" ");
        }

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [GetDomainSuggestions](#) na 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 [AWS Code Examples Repository](#).

```
public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
    try {
        ListPricesRequest pricesRequest = ListPricesRequest.builder()
            .tld(domainType)
            .build();

        ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
        listRes.stream()
            .flatMap(r -> r.prices().stream())
            .forEach(content -> System.out.println(" Name: " +
content.name() +
                " Registration: " + content.registrationPrice().price()
+ " "
                + content.registrationPrice().currency() +
                " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 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 [AWS Code Examples Repository](#).

```
public static void listDomains(Route53DomainsClient route53DomainsClient) {
    try {
        ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
        listRes.stream()
            .flatMap(r -> r.domains().stream())
            .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 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 [AWS Code Examples Repository](#).

```
public static void listOperations(Route53DomainsClient route53DomainsClient) {
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        localDateTime = localDateTime.minusYears(1);
        Instant myTime = localDateTime.toInstant(zoneOffset);

        ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
            .submittedSince(myTime)
            .build();

        ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
        listRes.stream()
            .flatMap(r -> r.operations().stream())
            .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                " Status: " + content.statusAsString() +
                " Date: " + content.submittedDate()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [ListOperations](#) na 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 [AWS Code Examples Repository](#).

```
public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
    String domainSuggestion,
    String phoneNumber,
    String email,
    String firstName,
    String lastName,
    String city) {

    try {
        ContactDetail contactDetail = ContactDetail.builder()
            .contactType(ContactType.COMPANY)
            .state("LA")
            .countryCode(CountryCode.IN)
            .email(email)
            .firstName(firstName)
            .lastName(lastName)
            .city(city)
            .phoneNumber(phoneNumber)
            .organizationName("My Org")
            .addressLine1("My Address")
            .zipCode("123 123")
            .build();

        RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
            .adminContact(contactDetail)
            .registrantContact(contactDetail)
            .techContact(contactDetail)
            .domainName(domainSuggestion)
            .autoRenew(true)
            .durationInYears(1)
            .build();
```

```
        RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
        System.out.println("Registration requested. Operation Id: " +
response.operationId());
        return response.operationId();

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obter detalhes da API, consulte [RegisterDomain](#) na Referência AWS SDK for Java 2.x da API.

Exibir faturamento

O exemplo de código a seguir mostra como visualizar registros de cobrança.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
    try {
        Date currentDate = new Date();
        LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
        ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
        LocalDateTime localDateTime2 = localDateTime.minusYears(1);
        Instant myStartTime = localDateTime2.toInstant(zoneOffset);
        Instant myEndTime = localDateTime.toInstant(zoneOffset);

        ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
```

```
        .start(myStartTime)
        .end(myEndTime)
        .build();

    ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
    listRes.stream()
        .flatMap(r -> r.billingRecords().stream())
        .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                " Operation: " + content.operationAsString() +
                " Price: " + content.price()));

    } catch (Route53Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte [ViewBilling](#) 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 = ""
```

```

Usage:
    <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

Where:
    domainType - The domain type (for example, com).\s
    phoneNumber - The phone number to use (for example,
+91.9966564xxx)    email - The email address to use.    domainSuggestion - The
domain suggestion (for example, findmy.accountants).\s
    firstName - The first name to use to register a domain.\s
    lastName - The last name to use to register a domain.\s
    city - the city to use to register a domain.\s
    """;

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)
    .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 da API do AWS SDK for Java 2.x .
 - [CheckDomainAvailability](#)
 - [CheckDomainTransferability](#)
 - [GetDomainDetail](#)

- [GetDomainSuggestions](#)
- [GetOperationDetail](#)
- [ListDomains](#)
- [ListOperations](#)
- [ListPrices](#)
- [RegisterDomain](#)
- [ViewBilling](#)

Exemplos do Amazon S3 usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * 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 HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte [ListBuckets](#) na Referência AWS SDK for Java 2.x da API.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * 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 S3Cors {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:
        <bucketName> <accountId>\s

    Where:
        bucketName - The Amazon S3 bucket to upload an object into.
        accountId - The id of the account that owns the Amazon S3
bucket.

    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String accountId = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setCorsInformation(s3, bucketName, accountId);
getBucketCorsInformation(s3, bucketName, accountId);
deleteBucketCorsInformation(s3, bucketName, accountId);
s3.close();
}

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](#) na 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 a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
```

```
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * 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 LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service
                (Amazon S3) bucket to upload an object into.
                accountId - The id of the account that owns the
                Amazon S3 bucket.

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
String accountId = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

setLifecycleConfig(s3, bucketName, accountId);
getLifecycleConfig(s3, bucketName, accountId);
deleteLifecycleConfig(s3, bucketName, accountId);
System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
s3.close();
}

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](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * 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 SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <polFile>
```

```

        Where:
            bucketName - The Amazon S3 bucket to set the policy on.
            polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String polFile = args[1];
    String policyText = getBucketPolicyFromFile(polFile);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setPolicy(s3, bucketName, policyText);
    s3.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

Copie um objeto usando um [S3Client](#).

```
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;

/**
 * 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 CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
                example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
                bucket2).

            """;

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

```

    }

    String objectKey = args[0];
    String fromBucket = args[1];
    String toBucket = args[2];
    System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

```
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.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * 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 CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The name of the bucket to create. The bucket name
                must be unique, or an error occurs.
                """;

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

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```



```
        createBucket(s3, bucketName);
        s3.close();
    }

    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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * 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 DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from (for
example, bucket1).""";

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

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // 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](#) a 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 [AWS Code Examples Repository](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- Para obter detalhes da API, consulte [DeleteBucket](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * 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 DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName>

                Where:
                    bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteBucketObjects(s3, bucketName);
    s3.close();
}

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](#) na 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 de 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * 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 DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
configuration from.
            """;

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

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    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 [AWS Code Examples Repository](#).

Determine o tipo de conteúdo de um objeto.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * 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 GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>>
```



```
        Where:
            bucketName - The Amazon S3 bucket name.\s
            keyName - The key name.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getContentType(s3, bucketName, keyName);
    s3.close();
}

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.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    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 [AWS Code Examples Repository](#).

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.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
```

```
import java.nio.file.Path;
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,
        URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
```

```
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;

public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket.\s
                topicArn - The Simple Notification Service topic ARN.\s
                id - An id value used for the topic configuration. This value is
displayed in the AWS Management Console.\s
            """;

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

        String bucketName = args[0];
        String topicArn = args[1];
        String id = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3Client = S3Client.builder()
            .region(region)
            .build();

        setBucketNotification(s3Client, bucketName, topicArn, id);
        s3Client.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

Leia dados como uma matriz de bytes usando um [S3Client](#).

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * 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 GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;
```

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

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    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.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
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](#).

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

```
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * 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 GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    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](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * 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 GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

```

```

        Usage:
            <bucketName> <keyName>\s

        Where:
            bucketName - The Amazon S3 bucket name.
            keyName - A key name that represents the object.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getURL(s3, bucketName, keyName);
    s3.close();
}

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](#).

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
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;

/**
 * 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 GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
```

```

        .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    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.setRequestProperty(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](#).

```

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * 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 GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = ""

        Usage:

```

```

        <bucketName> <keyName> <path>

        Where:
        bucketName - The Amazon S3 bucket name.\s
        keyName - The key name.\s
        path - The path where the file is written to.\s
        """;

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

    String bucketName = args[0];
    String keyName = args[1];
    String path = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getObjectBytes(s3, bucketName, keyName, path);
    s3.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * 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 GetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <objectKey>

            Where:
                bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
                objectKey - The object to get the ACL for.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    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 [GetBucketAcla](#) 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 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * 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 GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;

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

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    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](#) a Referência AWS SDK for Java 2.x da API.

Listar buckets

O exemplo de código a seguir mostra como listar buckets do S3.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * 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 ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
```

```

        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}

```

- Para obter detalhes da API, consulte [ListBuckets](#) 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 [AWS Code Examples Repository](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * 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 ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    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](#) da 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 [AWS Code Examples Repository](#).

```

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;

/**
 * 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 ListObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    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 " + calcKb(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 calcKb(Long val) {
        return val / 1024;
    }
}
```

Liste objetos usando paginação.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        listBucketObjects(s3, bucketName);
        s3.close();
    }

    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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 * For more information about restoring an object, see "Restoring an archived
 * object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * 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 RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value
of Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
            """;

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

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
```

```
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
    s3.close();
}

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 [AWS Code Examples Repository](#).

```
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Type;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

/**
 * 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 SetAcl {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <id>\s

                Where:
                bucketName - The Amazon S3 bucket to grant permissions on.\s
                id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
                """;

        if (args.length != 2) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String id = args[1];
    System.out.format("Setting access \n");
    System.out.println(" in bucket: " + bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setBucketAcl(s3, bucketName, id);
    System.out.println("Done!");
    s3.close();
}

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 de 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 [AWS Code Examples Repository](#).

```

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * 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 SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName> [indexdoc]\s

                Where:

```



```

        bucketName - The Amazon S3 bucket to set the website
configuration on.\s
        indexdoc - The index document, ex. 'index.html'
                    If not specified, 'index.html' will be set.
        """;

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

    String bucketName = args[0];
    String indexDoc = "index.html";
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setWebsiteConfig(s3, bucketName, indexDoc);
    s3.close();
}

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 carregar 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 [AWS Code Examples Repository](#).

Faça upload de um arquivo em um bucket usando um [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * 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 PutObject {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```

        <bucketName> <objectKey> <objectPath>\s

    Where:
        bucketName - The Amazon S3 bucket to upload an object into.
        objectKey - The object to upload (for example, book.pdf).
        objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
        """;

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

    String bucketName = args[0];
    String objectKey = args[1];
    String objectPath = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    putS3Object(s3, bucketName, objectKey, objectPath);
    s3.close();
}

// 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.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String bucketName,
                            String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .addTransferListener(LoggingTransferListener.create())
            .source(Paths.get(filePathURI))
            .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);
}
```

```
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return byteArray;
}
}
```

Faça upload de um objeto em um bucket e defina metadados usando um [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * 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 PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <objectKey> <objectPath>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

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

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // 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](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * 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 PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRetentionPeriod(s3, key, bucketName);
        s3.close();
    }

    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 [AWS Code Examples Repository](#).

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.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
        URI 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](#) na Referência AWS SDK for Java 2.x da API.

Use SQL com o Amazon S3 Select

O exemplo de código a seguir mostra como recuperar um subconjunto de dados usando SQL.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

O exemplo a seguir mostra uma consulta usando um objeto JSON. O [exemplo completo](#) também mostra o uso de um objeto CSV.

```
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.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
```

```
static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

public static void main(String[] args) {
    SelectObjectContentExample selectObjectContentExample = new
SelectObjectContentExample();
    try {
        SelectObjectContentExample.setUp();
        selectObjectContentExample.runSelectObjectContentMethodForJSON();
        selectObjectContentExample.runSelectObjectContentMethodForCSV();
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
        System.exit(1);
    } finally {
        SelectObjectContentExample.tearDown();
    }
}

EventStreamInfo runSelectObjectContentMethodForJSON() {
    // Set up request parameters.
    final String queryExpression = "select * from s3object[*][*] c where c.area
< 350000";
    final String fileType = FILE_JSON;

    InputSerialization inputSerialization = InputSerialization.builder()
        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
```

```

        // Call the selectObjectContent method with the request and a response
        handler.
        // Supply an EventStreamInfo object to the response handler to gather
        records and information from the response.
        s3AsyncClient.selectObjectContent(select,
        buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
            record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times", fileType,
        eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
        eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
        eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
    eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
        information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
        SelectObjectContentResponseHandler.Visitor.builder()
            .onRecords(r -> {
                logger.info("Record event received.");
                eventStreamInfo.addRecord(r.payload().asUtf8String());
                eventStreamInfo.incrementOnRecordsCalled();
            })
            .onCont(ce -> {
                logger.info("Continuation event received.");
                eventStreamInfo.incrementContinuationEvents();
            })
            .onProgress(pe -> {
                Progress progress = pe.details();
                logger.info("Progress event received:\n bytesScanned:
        {} \n bytesProcessed: {} \n bytesReturned: {}",
                    progress.bytesScanned(),
                    progress.bytesProcessed(),
                    progress.bytesReturned());
            })
    }

```

```
        .onEnd(ee -> logger.info("End event received."))
        .onStats(se -> {
            logger.info("Stats event received.");
            eventStreamInfo.addStats(se.details());
        })
        .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
    // processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
// processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }
}
```



```
    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- Para obter detalhes da API, consulte [SelectObjectContent](#) na 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 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 [AWS Code Examples Repository](#).

Gere um URL pré-assinado para um objeto e faça download dele (solicitação GET).

Importações.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
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.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Gere o URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL will
                expire in 10 minutes.
                .getObjectRequest(objectRequest)
                .build();

        PresignedGetObjectRequest presignedRequest =
            presigner.presignGetObject(presignRequest);
```

```

        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Faça o download do objeto usando uma das três abordagens a seguir.

Use a classe `HttpURLConnection` do JDK (desde a v1.1) para fazer o download.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Use a classe `HttpClient` do JDK (desde a v11) para fazer o download.

```

/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

```

```

HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
HttpClient httpClient = HttpClient.newHttpClient();
try {
    URL presignedUrl = new URL(presignedUrlString);
    HttpResponse<InputStream> response = httpClient.send(requestBuilder
        .uri(presignedUrl.toURI())
        .GET()
        .build(),
        HttpResponse.BodyHandlers.ofInputStream());

    IoUtils.copy(response.body(), byteArrayOutputStream);

    logger.info("HTTP response code is " + response.statusCode());
} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}

```

Use a classe AWS SDK for SdkHttpClient Java para fazer o download.

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
            sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(

```

```

        abortableInputStream -> {
            try {
                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        },
        () -> logger.error("No response body."));

        logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
    }
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}

```

Gere um URL atribuído previamente para um upload e faça o upload de um arquivo (solicitação PUT).

Importações.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
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;
```

Gere o URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

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

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires
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("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Faça upload de um objeto de arquivo usando uma das três abordagens a seguir.

Use a classe `HttpURLConnection` do JDK (desde a v1.1) para fazer o upload.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        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 a classe `HttpClient` do JDK (desde a v11) para fazer o upload.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    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(new URL(presignedUrlString).toURI())
            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Use a `SdkHttpClient` classe AWS for Java V2 para fazer o upload.

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);
    }
}

```



```
        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
v));

        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

Conceitos básicos de buckets e objetos

O código de exemplo a seguir mostra como:

- Criar um bucket e fazer 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.
- Excluir 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 = ""

            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The Amazon S3 bucket to create.
                key - The key to use.
                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).\s
        """;

    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];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .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. Upload 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 da 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 [AWS Code Examples Repository](#).

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 [AWS Code Examples Repository](#).

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 da 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 download de 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 [AWS Code Examples Repository](#).

Chame funções que transferem arquivos de e para um bucket do S3 usando o TransferManager S3.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .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,
    URI 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, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create())
        .source(Paths.get(filePathURI))
        .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 [AWS Code Examples Repository](#).

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 s3CrtAsyncClient - 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 s3CrtAsyncClient,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s3CrtAsyncClient.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 [AWS Code Examples Repository](#).

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 `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 da 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 do 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);
}
}
```

Exemplos do S3 Glacier usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 vault

O exemplo de código a seguir mostra como criar um vault (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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * 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 CreateVault {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to create.

            """;

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

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;
```

```
/**
 * 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 DeleteVault {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName>

            Where:
                vaultName - The name of the vault to delete.\s
            """;

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

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * 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 DeleteArchive {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
```

```

        vaultName - The name of the vault that contains the archive to
delete.

        accountId - The account ID value.
        archiveId - The archive ID value.
        """;

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

    String vaultName = args[0];
    String accountId = args[1];
    String archiveId = args[2];
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
    glacier.close();
}

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 archive was deleted.");

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

```

- Para obter detalhes da API, consulte [DeleteArchive](#) a Referência AWS SDK for Java 2.x da API.

Liste cofres

O exemplo de código a seguir mostra como listar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * 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 ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllVault(glacier);
        glacier.close();
    }

    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](#) Referência AWS SDK for Java 2.x da API.

Recuperar um inventário do cofre

O exemplo de código a seguir mostra como recuperar um inventário de 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * 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 ArchiveDownload {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <vaultName> <accountId> <path>
```

```
        Where:
            vaultName - The name of the vault.
            accountId - The account ID value.
            path - The path where the file is written to.
        """;

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

    String vaultName = args[0];
    String accountId = args[1];
    String path = args[2];
    GlacierClient glacier = GlacierClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String jobNum = createJob(glacier, vaultName, accountId);
    checkJob(glacier, jobNum, vaultName, accountId, path);
    glacier.close();
}

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.

Fazer upload de um arquivo em um cofre

O exemplo de código a seguir mostra como fazer upload de um arquivo em 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
```

```
import java.security.NoSuchAlgorithmException;

/**
 * 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 UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <strPath> <vaultName>\s

            Where:
                strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                vaultName - The name of the vault.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }
}
```

```
public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
        UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
            .vaultName(vaultName)
            .checksum(checkVal)
            .build();

        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();

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

private static String computeSHA256(File inputFile) {
    try {
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
NoSuchAlgorithmException {

    byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
    return computeSHA256TreeHash(chunkSHA256Hashes);
}
```

```
}

/**
 * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
 * includes the checksum for the last chunk, even if it's smaller than 1 MB.
 */
public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
    NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    long numChunks = file.length() / ONE_MB;
    if (file.length() % ONE_MB > 0) {
        numChunks++;
    }

    if (numChunks == 0) {
        return new byte[][] { md.digest() };
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;

    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
file.getName(),
```



```
        ioe.getMessage());
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

            // If there are at least two elements remaining.
            if (prevLvlHashes.length - i > 1) {

                // Calculate a digest of the concatenated nodes.
                md.reset();
                md.update(prevLvlHashes[i]);
                md.update(prevLvlHashes[i + 1]);
                currLvlHashes[j] = md.digest();

            } else { // Take care of the remaining odd chunk
                currLvlHashes[j] = prevLvlHashes[i];
            }
        }

        prevLvlHashes = currLvlHashes;
    }

    return prevLvlHashes[0];
}
```

```
/**
 * Returns the hexadecimal representation of the input byte array
 */
public static String toHex(byte[] data) {
    StringBuilder sb = new StringBuilder(data.length * 2);
    for (byte datum : data) {
        String hex = Integer.toHexString(datum & 0xFF);

        if (hex.length() == 1) {
            // Append leading zero.
            sb.append("0");
        }
        sb.append(hex);
    }
    return sb.toString().toLowerCase();
}
}
```

- Para obter detalhes da API, consulte [UploadArchive](#) 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
            .region(region)
            .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

Criar 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 da API do AWS SDK for Java 2.x .
 - [CreatePipeline](#)
 - [UpdatePipeline](#)

Excluir 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 [AWS Code Examples Repository](#).

```
// 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](#) Referência AWS SDK for Java 2.x da API.

Descrever uma execução de 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 [AWS Code Examples Repository](#).

```
// 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](#) na Referência AWS SDK for Java 2.x da API.

Executar 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 [AWS Code Examples Repository](#).

```

// 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

Iniciar com trabalhos geoespaciais e pipelines

O código de exemplo a seguir mostra como:

- Configurar recursos para um pipeline.
- Configurar um pipeline que execute um trabalho geoespacial.
- Inicie a execução de um pipeline.
- Monitorar o status da execução.
- Ver 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\":
{\"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 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 da API do AWS SDK for Java 2.x .
 - [CreatePipeline](#)
 - [DeletePipeline](#)
 - [DescribePipelineExecution](#)
 - [StartPipelineExecution](#)
 - [UpdatePipeline](#)

Exemplos do Secrets Manager usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.CreateSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.CreateSecretResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * 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 CreateSecret {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <secretName> <secretValue>\s

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
```



```

        secretValue - The secret value.\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    String secretValue = args[1];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    String secretARN = createNewSecret(secretsClient, secretName, secretValue);
    System.out.println("The secret ARN is " + secretARN);
    secretsClient.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.DeleteSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * 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 DeleteSecret {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <secretName>\s

            Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

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

```
String secretName = args[0];
Region region = Region.US_EAST_1;
SecretsManagerClient secretsClient = SecretsManagerClient.builder()
    .region(region)
    .build();

deleteSpecificSecret(secretsClient, secretName);
secretsClient.close();
}

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 [DeleteSecret](#) na Referência AWS SDK for Java 2.x da API.

Descrever 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.DescribeSecretRequest;
import software.amazon.awssdk.services.secretsmanager.model.DescribeSecretResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * 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 DescribeSecret {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName>\s

                Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

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

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        describeGivenSecret(secretsClient, secretName);
    }
}
```

```
        secretsClient.close();
    }

    public static void describeGivenSecret(SecretsManagerClient secretsClient,
String secretName) {
        try {
            DescribeSecretRequest secretRequest = DescribeSecretRequest.builder()
                .secretId(secretName)
                .build();

            DescribeSecretResponse secretResponse =
secretsClient.describeSecret(secretRequest);
            Instant lastChangedDate = secretResponse.lastChangedDate();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());

            formatter.format(lastChangedDate);
            System.out.println("The date of the last change to " +
secretResponse.name() + " is " + lastChangedDate);

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

- Para obter detalhes da API, consulte [DescribeSecreta](#) Referência AWS SDK for Java 2.x da API.

Obter 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * 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
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-
secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <secretName>\s

                Where:
                secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;
```

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

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    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 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.ListSecretsResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretListEntry;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import java.util.List;

/**
 * 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 ListSecrets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
            .region(region)
            .build();

        listAllSecrets(secretsClient);
        secretsClient.close();
    }

    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](#) Referência AWS SDK for Java 2.x da API.

Modificar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;
import software.amazon.awssdk.services.secretsmanager.model.UpdateSecretRequest;

/**
 * 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 UpdateSecret {
    public static void main(String[] args) {
        final String usage = ""
```

```

        Usage:
            <secretName> <secretValue>

        Where:
            secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
            secretValue - The secret value that is updated.\s
            """;

    if (args.length < 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String secretName = args[0];
    String secretValue = args[1];
    Region region = Region.US_EAST_1;
    SecretsManagerClient secretsClient = SecretsManagerClient.builder()
        .region(region)
        .build();

    updateMySecret(secretsClient, secretName, secretValue);
    secretsClient.close();
}

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.

Inserir um valor em um segredo

O exemplo de código a seguir mostra como inserir 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.PutSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * 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 PutSecret {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <secretName> <secretValue>

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                    secretValue - The text to encrypt and store in the new version
of the secret.\s
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String secretName = args[0];
String secretValue = args[1];
Region region = Region.US_EAST_1;
SecretsManagerClient secretsClient = SecretsManagerClient.builder()
    .region(region)
    .build();

putSecret(secretsClient, secretName, secretValue);
secretsClient.close();
}

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 o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        listAllTemplates(sesv2Client);
    }
}
```

```
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 [List Templates](#) a Referência AWS SDK for Java 2.x da API.

Listar identidades

O exemplo de código a seguir mostra como listar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;
```

```
/**
 * 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 ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
            .region(region)
            .build();

        listSESIIdentities(client);
    }

    public static void listSESIIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }
        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte [ListIdentities](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * 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 SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s

            """;

        if (args.length != 3) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String sender = args[0];
    String recipient = args[1];
    String subject = args[2];

    Region region = Region.US_EAST_1;
    SesClient client = SesClient.builder()
        .region(region)
        .build();

    // The HTML body of the email.
    String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
        + "<p> See the list of customers.</p>" + "</body>" + "</html>";

    try {
        send(client, sender, recipient, subject, bodyHTML);
        client.close();
        System.out.println("Done");
    } catch (MessagingException e) {
        e.printStackTrace();
    }
}

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);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
```

```
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * 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 SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject> <fileLocation>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
                fileLocation - The location of a Microsoft Excel file to use as
an attachment (C:/AWS/customers.xls).\s
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
            + "of customers to contact.";
```

```
// The HTML body of the email.
String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
    + "<p>Please see the attached file for a " + "list of customers to
contact.</p>" + "</body>"
    + "</html>";

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

try {
    sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
    client.close();
    System.out.println("Done");
} catch (IOException | MessagingException e) {
    e.printStackTrace();
}
}

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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) 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
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <template> <sender> <recipient>\s

            Where:
                template - The name of the email template.
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s

        """;
```

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

    String templateName = args[0];
    String sender = args[1];
    String recipient = args[2];
    Region region = Region.US_EAST_1;
    SesV2Client sesv2Client = SesV2Client.builder()
        .region(region)
        .build();

    send(sesv2Client, sender, recipient, templateName);
}

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 de API v2 do Amazon SES usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

Envia uma mensagem.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) 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 SendEmail {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <sender> <recipient> <subject>\s

                Where:
```

```
sender.\s                sender - An email address that represents the
recipient.\s            recipient - An email address that represents the
                        subject - The subject line.\s
                        """;

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

    String sender = args[0];
    String recipient = args[1];
    String subject = args[2];

    Region region = Region.US_EAST_1;
    SesV2Client sesv2Client = SesV2Client.builder()
        .region(region)
        .build();

    // The HTML body of the email.
    String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
        + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

    send(sesv2Client, sender, recipient, subject, bodyHTML);
}

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](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon SNS usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
package com.example.sns;

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)
            .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](#) na Referência AWS SDK for Java 2.x da API.

Tópicos

- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

Ações

Adicionar tags a um tópico

O exemplo de código a seguir mostra como adicionar tags a 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 [AWS Code Examples Repository](#).

```
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.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * 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 AddTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to which tags are added.

            """;

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

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
    software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * 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 CheckOptOut {
    public static void main(String[] args) {

        final String usage = ""

            Usage:    <phoneNumber>

            Where:
                phoneNumber - The mobile phone number to look up (for example,
+1XXX5550100).

            """;

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

```
String phoneNumber = args[0];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

checkPhone(snsClient, phoneNumber);
snsClient.close();
}

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 para o endpoint por uma ação `Subscribe` anterior.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * 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 ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <subscriptionToken> <topicArn>

                Where:
                    subscriptionToken - A short-lived token sent to an endpoint
during the Subscribe action.
                    topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    confirmSub(snsClient, subscriptionToken, topicArn);
    snsClient.close();
}

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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            """;

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

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    String arnVal = createSNSTopic(snsClient, topicName);
    System.out.println("The topic ARN is" + arnVal);
    snsClient.close();
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    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 códigos 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of the subscription to delete.
            """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 DeleteTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to delete.
            """;

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

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    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 [DeleteTópica](#) 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * 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 GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println("Getting attributes for a topic with name: " + topicArn);
    getSnsTopicAttributes(snsClient, topicArn);
    snsClient.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * 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 GetSMSAttributes {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn>

                Where:
                    topicArn - The ARN of the topic from which to retrieve
attributes.

        """;

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

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    getSNSAttrutes(snsClient, topicArn);
    snsClient.close();
}

public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
    try {
        GetSubscriptionAttributesRequest request =
        GetSubscriptionAttributesRequest.builder()
            .subscriptionArn(topicArn)
            .build();

        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
        snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();

        // Iterate through the map
        Iterator iter = map.entrySet().iterator();
        while (iter.hasNext()) {
            Map.Entry entry = (Map.Entry) iter.next();
            System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
            entry.getValue());
        }

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

    System.out.println("\n\nStatus was good");
}
}
```

- Para obter detalhes da API, consulte [GetSMSAttributes](#) na Referência da 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * 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 ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
                ListPhoneNumbersOptedOutRequest.builder().build();
```

```

        ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
        System.out.println("Status is " + result.sdkHttpResponse().statusCode()
+ "\n\nPhone Numbers: \n\n"
                + result.phoneNumbers());

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

```

- Para obter detalhes da API, consulte [ListPhoneNumbersOptedOut](#) 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 [AWS Code Examples Repository](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * 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 ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    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\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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

        Usage:    <message> <phoneNumber>
```

```

        Where:
            message - The message text to send.
            phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String phoneNumber = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTextSMS(snsClient, message, phoneNumber);
    snsClient.close();
}

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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 PublishTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <topicArn>

            Where:
                message - The message text to send.
                topicArn - The ARN of the topic to publish.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
```

```
String topicArn = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();
pubTopic(snsClient, message, topicArn);
snsClient.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * 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 UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of a subscription.

            "";

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

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
        }
    }
}
```

```
fp.addAttribute("event", "order_placed");

// Add a prefix attribute
fp.addAttributePrefix("customer_interests", "bas");

// Add an anything-but attribute
fp.addAttributeAnythingBut("customer_interests", "baseball");

// Add a filter policy attribute with a list of values
ArrayList<String> attributeValues = new ArrayList<>();
attributeValues.add("rugby");
attributeValues.add("soccer");
attributeValues.add("hockey");
fp.addAttribute("customer_interests", attributeValues);

// Add a numeric attribute
fp.addAttribute("price_usd", "=", 0);

// Add a numeric attribute with a range
fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

// Apply the filter policy attributes to an Amazon SNS subscription
fp.apply(snsClient, subscriptionArn);

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

- 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * 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 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)
            .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 para 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * 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 SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.
            """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    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](#) a 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 inscrever uma função do Lambda para receber 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 [AWS Code Examples Repository](#).

```

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;

/**
 * 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 SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    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 inscrever um endpoint HTTP ou HTTPS para receber 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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 SubscribeHTTPS {
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:    <topicArn> <url>

    Where:
        topicArn - The ARN of the topic to subscribe.
        url - The HTTPS endpoint that you want to receive notifications.
    """;

if (args.length < 2) {
    System.out.println(usage);
    System.exit(1);
}

String topicArn = args[0];
String url = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

subHTTPS(snsClient, topicArn, url);
snsClient.close();
}

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 [AWS Code Examples Repository](#).

```
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;

/**
 * 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 SubscribeEmail {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <topicArn> <email>

            Where:
                topicArn - The ARN of the topic to subscribe.
                email - The email address to use.
            """;
```

```
    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    String email = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    subEmail(snsClient, topicArn, email);
    snsClient.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * 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, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:    <token> <platformApplicationArn>
```

```
Where:
```

```
token - The name of the FIFO topic.\s
```

```
platformApplicationArn - The ARN value of platform application.
```

```
You can get this value from the AWS Management Console.\s
```

```
""";
```

```
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)
    .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

O exemplo de código a seguir mostra 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 [AWS Code Examples Repository](#).

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)
    .messageDeduplicationId(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 da API do AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Publicar](#)
 - [Assinar](#)

Publicar mensagens SMS em um tópico

O código de exemplo 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 [AWS Code Examples Repository](#).

Criar um tópico e retorne seu ARN.

```
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;

/**
 * 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 CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicName>

                Where:
                    topicName - The name of the topic to create (for example,
mytopic).

                """;
```

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

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        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.

```
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;
```



```
/**
 * 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 SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <topicArn> <phoneNumber>

                Where:
                    topicArn - The ARN of the topic to subscribe.
                    phoneNumber - A mobile phone number that receives notifications
(for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    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.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * 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 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)
            .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);
        }
    }
}

```

Publique uma mensagem em um tópico. A mensagem é enviada para todos os assinantes.

```

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;

/**
 * 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 PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

```

```
        Where:
            message - The message text to send.
            phoneNumber - The mobile phone number to which a message is sent
(for example, +1XXX5550100).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String message = args[0];
    String phoneNumber = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTextSMS(snsClient, message, phoneNumber);
    snsClient.close();
}

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 sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

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 SNS com o Lambda usando Java.

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
    }
}
```

```
    }  
    return Boolean.TRUE;  
}  
  
public void processRecord(SNSRecord record) {  
    try {  
        String message = record.getSNS().getMessage();  
        logger.log("message: " + message);  
    } catch (Exception e) {  
        throw new RuntimeException(e);  
    }  
}  
}
```

Exemplos do Amazon SQS usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
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)
            .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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sqs.SqsClient;  
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;  
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;  
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;  
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;  
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;  
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;  
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;  
import software.amazon.awssdk.services.sqs.model.Message;  
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;  
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;  
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;  
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
```



```
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * 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 SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient
                .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        } catch (SqsException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

```
        return getQueueUrlResponse.queueUrl();

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

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

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

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
    System.out.println("Queue URLs with prefix: " + namePrefix);
    for (String url : listQueuesFilteredResponse.queueUrls()) {
        System.out.println(url);
    }

    System.out.println("\nSend message");
}
```

```
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello world!")
                .delaySeconds(10)
                .build());
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                .queueUrl(queueUrl)

                .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)

                    .build())

                .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
```

```
        .numberOfMessages(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 [CreateQueue](#) a 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 [AWS Code Examples Repository](#).

```
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 [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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * 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 DeleteQueue {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <queueName>

            Where:
                queueName - The name of the Amazon SQS queue to delete.

            """;

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

        String queueName = args[0];
```

```
SqsClient sqs = SqsClient.builder()
    .region(Region.US_WEST_2)
    .build();

deleteSQSQueue(sqs, queueName);
sqs.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- Para obter detalhes da API, consulte [GetQueueUrl](#) na 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 [AWS Code Examples Repository](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }

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

- Para obter detalhes da API, consulte [ListQueues](#) na Referência AWS SDK for Java 2.x da API.

Receba 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 [AWS Code Examples Repository](#).

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .numberOfMessages(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](#) a Referência AWS SDK for Java 2.x da API.

Para enviar mensagens em lote a 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 [AWS Code Examples Repository](#).

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

    .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),


SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
        .build())
    .build();
sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Para obter detalhes da API, consulte [SendMessageBatch](#) na Referência AWS SDK for Java 2.x da API.

Envie 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * 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 SendMessages {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <queueName> <message>

            Where:
                queueName - The name of the queue.
                message - The message to send.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Cenários

Criar e publicar em um tópico FIFO

O exemplo de código a seguir mostra 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 [AWS Code Examples Repository](#).

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)
```

```
        .messageDeduplicationId(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 da API do AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Publicar](#)
 - [Assinar](#)

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

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 SQS com o Lambda usando JavaScript.


```
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.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
    }
}
```

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 do Lambda que recebem eventos de uma fila do 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 do Step Functions usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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, 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 [AWS Code Examples Repository](#).

Versão Java do Olá.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * 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 ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    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](#) a 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 do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

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

- Para obter detalhes da API, consulte [CreateStateMachine](#) na Referência AWS SDK for Java 2.x da API.

Criar uma atividade

O exemplo de código a seguir mostra como criar uma atividade do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

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

- Para obter detalhes da API, consulte [CreateActivity](#) a 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 [AWS Code Examples Repository](#).

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }


    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}
```

- Para obter detalhes da API, consulte [DeleteStateMachine](#) a Referência AWS SDK for Java 2.x da API.

Excluir uma atividade.

Os exemplos de código a seguir mostram como excluir uma atividade do Step Functions.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);


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

- Para obter detalhes da API, consulte [DeleteActivity](#) a Referência AWS SDK for Java 2.x da API.

Descrever uma máquina de estado

O exemplo de código a seguir mostra como descrever 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 [AWS Code Examples Repository](#).


```
public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());


    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}
```

- Para obter detalhes da API, consulte [DescribeStateMachine](#) na Referência AWS SDK for Java 2.x da API.

Descrever a execução de uma máquina de estado

O exemplo de código a seguir mostra como descrever uma execução de 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 [AWS Code Examples Repository](#).

```
public static void describeExe(SfnClient sfnClient, String executionArn) {
```

```
try {
    DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
        .executionArn(executionArn)
        .build();

    String status = "";
    boolean hasSucceeded = false;
    while (!hasSucceeded) {
        DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
        status = response.statusAsString();
        if (status.compareTo("RUNNING") == 0) {
            System.out.println("The state machine is still running, let's
wait for it to finish.");
            Thread.sleep(2000);
        } else if (status.compareTo("SUCCEEDED") == 0) {
            System.out.println("The Step Function workflow has succeeded");
            hasSucceeded = true;
        } else {
            System.out.println("The Status is neither running or
succeeded");
        }
    }
    System.out.println("The Status is " + status);

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

- 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 tarefa 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 [AWS Code Examples Repository](#).

```
public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
    GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
    sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

/// <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](#) na Referência AWS SDK for Java 2.x da API.

Listar atividades

O exemplo de código a seguir mostra como listar 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * 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 ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Listar execuções de máquinas de estado

O exemplo de código a seguir mostra como listar execuções de 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 [AWS Code Examples Repository](#).

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
        .executionArn(exeARN)
        .maxResults(10)
        .build();
```

```
        GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event : events) {
            System.out.println("The event type is " + event.type().toString());
        }

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

- Para obter detalhes da API, consulte [ListExecutions](#) na Referência AWS SDK for Java 2.x da API.

Listar máquinas de estado

O exemplo de código a seguir mostra como listar estado de máquinas do Step Functions.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * 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 ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    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](#) na 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 de sucesso para uma tarefa de Step Functions.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);


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

- Para obter detalhes da API, consulte [SendTaskSuccess](#) Referência AWS SDK for Java 2.x da API.

Iniciar execução de máquina de estado

O exemplo de código a seguir mostra como iniciar uma execução de 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 [AWS Code Examples Repository](#).


```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

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

- Para obter detalhes da API, consulte [StartExecution](#) Referência AWS SDK for Java 2.x da API.

Cenários

Começar a usar máquinas de estado

O código de exemplo a seguir mostra como:

- Criar uma atividade.
- Criar uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Executar a máquina de estado e responder à atividade com entrada do usuário.
- Obter o status e a saída finais após a conclusão da execução e, em seguida, limpar 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 = ""

            Usage:
                <roleARN> <activityName> <stateMachineName>
```

```

        Where:
            roleName - The name of the IAM role to create for this state
machine.
            activityName - The name of an activity to create.
            stateMachineName - The name of the state machine to create.
        """;

    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)
        .build();

    Region regionGl = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(regionGl)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Step Functions example scenario.");

```

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

System.out.println(DASHES);
System.out.println("1. Create an activity.");
String activityArn = createActivity(sfnClient, activityName);
System.out.println("The ARN of the activity is " + activityArn);
System.out.println(DASHES);

// Get JSON to use for the state machine and place the activityArn value
into
// it.
InputStream input = StepFunctionsScenario.class.getClassLoader()
    .getResourceAsStream("chat_sfn_state_machine.json");
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
String jsonString = mapper.writeValueAsString(jsonNode);

// Modify the Resource node.
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);
((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

// Convert the modified Java object back to a JSON string.
String stateDefinition = objectMapper.writeValueAsString(root);
System.out.println(stateDefinition);

System.out.println(DASHES);
System.out.println("2. Create a state machine.");
String roleARN = createIAMRole(iam, roleName, polJSON);
String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
System.out.println("The ARN of the state machine is " + stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe the state machine.");
describeStateMachine(sfnClient, stateMachineArn);
System.out.println("What should ChatSFN call you?");
String userName = sc.nextLine();
System.out.println("Hello " + userName);
System.out.println(DASHES);

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

```
// The JSON to pass to the StartExecution call.
String executionJson = "{ \"name\" : \"" + userName + "\" }";
System.out.println(executionJson);
System.out.println("4. Start execution of the state machine and interact
with it.");
String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
System.out.println("The ARN of the state machine execution is " + runArn);
List<String> myList;
while (!action) {
    myList = getActivityTask(sfnClient, activityArn);
    System.out.println("ChatSFN: " + myList.get(1));
    System.out.println(userName + " please specify a value.");
    String myAction = sc.nextLine();
    if (myAction.compareTo("done") == 0)
        action = true;

    System.out.println("You have selected " + myAction);
    String taskJson = "{ \"action\" : \"" + myAction + "\" }";
    System.out.println(taskJson);
    sendTaskSuccess(sfnClient, myList.get(0), taskJson);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the execution.");
describeExe(sfnClient, runArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Delete the activity.");
deleteActivity(sfnClient, activityArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the state machines.");
deleteMachine(sfnClient, stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Step Functions example scenario is complete.");
System.out.println(DASHES);
}
```

```
public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        return response.role().arn();

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

public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
            if (status.compareTo("RUNNING") == 0) {
                System.out.println("The state machine is still running, let's
wait for it to finish.");
                Thread.sleep(2000);
            } else if (status.compareTo("SUCCEEDED") == 0) {
                System.out.println("The Step Function workflow has succeeded");
                hasSucceeded = true;
            } else {
                System.out.println("The Status is neither running or
succeeded");
            }
        }
    }
}
```

```
        System.out.println("The Status is " + status);

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

public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

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

public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();
```

```
sfnClient.deleteActivity(activityRequest);
System.out.println("You have deleted " + actArn);

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

public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());

    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}

public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();
```



```
        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}

public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

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

public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
```

```
        .roleArn(roleARN)
        .type(StateMachineType.STANDARD)
        .build();

    CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
    return response.stateMachineArn();

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

public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 *   "Version": "2012-10-17",
 *   "Statement": [
 *     {
 *       "Effect": "Allow",
 *       "Principal": {
 *         "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 *       },
 *       "Action": "sts:AssumeRole"
 *     }
 *   ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *

```

```
* 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 AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <roleArn> <roleSessionName>\s

            Where:
                roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
                roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
            .region(region)
            .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */
```

```
public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
                DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
                supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```



```
}
```

- 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 [AWS Code Examples Repository](#).

```
public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
    }
}
```

```
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

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

- 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 [AWS Code Examples Repository](#).

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
        AddAttachmentsToSetRequest.builder()
```

```
        .attachments(attachment)
        .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

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

- 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 [AWS Code Examples Repository](#).

```
public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
```

```
        .language("en")
        .issueType("technical")
        .build();

    CreateCaseResponse response = supportClient.createCase(caseRequest);
    return response.caseId();

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

- Para obter detalhes da API, consulte [CreateCase](#) 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 caso do AWS Support .

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
        .attachmentId(attachId)
        .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());
    }
```

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

- Para obter detalhes da API, consulte [DescribeAttachment](#) 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 [AWS Code Examples Repository](#).

```
public static void getOpenCase(SupportClient supportClient) {  
    try {  
        // Specify the start and end time.  
        Instant now = Instant.now();  
        java.time.LocalDate.now();  
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);  
  
        DescribeCasesRequest describeCasesRequest =  
DescribeCasesRequest.builder()  
            .maxResults(20)  
            .afterTime(yesterday.toString())  
            .beforeTime(now.toString())  
            .build();  
  
        DescribeCasesResponse response =  
supportClient.describeCases(describeCasesRequest);  
        List<CaseDetails> cases = response.cases();  
        for (CaseDetails sinCase : cases) {  
            System.out.println("The case status is " + sinCase.status());  
            System.out.println("The case Id is " + sinCase.caseId());  
        }  
    }  
}
```

```
        System.out.println("The case subject is " + sinCase.subject());
    }

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

- Para obter detalhes da API, consulte [DescribeCases](#) a 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 [AWS Code Examples Repository](#).

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm : communications) {
            System.out.println("the body is: " + comm.body());

            // Get the attachment id value.
```

```
        List<AttachmentDetails> attachments = comm.attachmentSet();
        for (AttachmentDetails detail : attachments) {
            attachId = detail.attachmentId();
        }
    }
    return attachId;

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

- Para obter detalhes da API, consulte [DescribeCommunications](#) na 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 [AWS Code Examples Repository](#).

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
```

```

String catName = null;
List<String> sevCatList = new ArrayList<>();
List<Service> services = response.services();

System.out.println("Get the first 10 services");
int index = 1;
for (Service service : services) {
    if (index == 11)
        break;

    System.out.println("The Service name is: " + service.name());
    if (service.name().compareTo("Account") == 0)
        serviceCode = service.code();

    // Get the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat : categories) {
        System.out.println("The category name is: " + cat.name());
        if (cat.name().compareTo("Security") == 0)
            catName = cat.name();
    }
    index++;
}

// Push the two values to the list.
sevCatList.add(serviceCode);
sevCatList.add(catName);
return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}

```

- 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 [AWS Code Examples Repository](#).

```
public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

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

- Para obter detalhes da API, consulte [DescribeSeverityLevels](#) na 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 [AWS Code Examples Repository](#).

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

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

- Para obter detalhes da API, consulte [ResolveCase](#) 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 [AWS Code Examples Repository](#).

Execute várias AWS Support operações.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
```

```
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * 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 = ""

```

```
Usage:
  <fileAttachment>Where:
  fileAttachment - The file can be a simple saved .txt file to use
as an email attachment.\s
  """;

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 yesterday = now.toLocalDate().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 da API do AWS SDK for Java 2.x .
 - [AddAttachmentsToSet](#)
 - [AddCommunicationToCase](#)
 - [CreateCase](#)
 - [DescribeAttachment](#)
 - [DescribeCases](#)
 - [DescribeCommunications](#)
 - [DescribeServices](#)
 - [DescribeSeverityLevels](#)
 - [ResolveCase](#)

Exemplos do Systems Manager usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <paraName>

            Where:
                paraName - The name of the parameter.
                paraValue - The value of the parameter.
```

```
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String paraName = args[0];
    String paraValue = args[1];
    Region region = Region.US_EAST_1;
    SsmClient ssmClient = SsmClient.builder()
        .region(region)
        .build();

    putParaValue(ssmClient, paraName, paraValue);
    ssmClient.close();
}

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.CreateOpsItemResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * 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 CreateOpsItem {
    public static void main(String[] args) {

        final String USAGE = ""

            Usage:
                <title> <source> <category> <severity>

            Where:
                title - The OpsItem title.
                source - The origin of the OpsItem, such as Amazon EC2 or AWS
Systems Manager.
                category - A category to assign to an OpsItem.
                severity - A severity to assign to an OpsItem.

            """;

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```



```
String title = args[0];
String source = args[1];
String category = args[2];
String severity = args[3];

Region region = Region.US_EAST_1;
SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

System.out
    .println("The Id of the OpsItem is " + createNewOpsItem(ssmClient,
title, source, category, severity));
    ssmClient.close();
}

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](#) a 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsRequest;
import software.amazon.awssdk.services.ssm.model.DescribeOpsItemsResponse;
import software.amazon.awssdk.services.ssm.model.OpsItemSummary;
import software.amazon.awssdk.services.ssm.model.SsmException;
import java.util.List;

/**
 * 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 DescribeOpsItems {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        describeItems(ssmClient);
    }
}
```

```
        ssmClient.close();
    }

    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.

Obter informações de parâmetros

O exemplo de código a seguir mostra como obter informações de 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
```

```
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * 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 GetParameter {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <paraName>

                Where:
                paraName - The name of the parameter.
                """;

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

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.SsmException;
import software.amazon.awssdk.services.ssm.model.UpdateOpsItemRequest;
import software.amazon.awssdk.services.ssm.model.OpsItemStatus;

/**
 * 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 ResolveOpsItem {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <opsID>

            Where:
                opsID - The Ops item ID value.
            """;

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

        String opsID = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();
        setOpsItemStatus(ssmClient, opsID);
    }

    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](#) na Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Textract usando o SDK para 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, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * 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 AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sourceDoc>\s

            Where:
                sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
                """;

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

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
            .region(region)
            .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    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](#) na 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 [AWS Code Examples Repository](#).

Detectar texto de um documento de entrada.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * 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 DetectDocumentText {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
```

```
<sourceDoc>\s

    Where:
        sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
        """;

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

    String sourceDoc = args[0];
    Region region = Region.US_EAST_2;
    TextractClient textractClient = TextractClient.builder()
        .region(region)
        .build();

    detectDocText(textractClient, sourceDoc);
    textractClient.close();
}

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);
    }
}
}

```

Detectar texto de um documento localizado em um bucket do Amazon S3.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * 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 DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = ""

        Usage:

```

```

        <bucketName> <docName>\s

        Where:
            bucketName - The name of the Amazon S3 bucket that contains the
document.\s

            docName - The document name (must be an image, i.e., book.png).
\s

        """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
            .region(region)
            .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }

    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 =
textextractClient.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 [DetectDocumentText](#) Referência AWS SDK for Java 2.x da API.

Iniciar análise assíncrona de um documento

O exemplo de código a seguir mostra como iniciar uma 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 [AWS Code Examples Repository](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
```

```
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * 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 StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <docName>\s

                Where:
                bucketName - The name of the Amazon S3 bucket that contains the
document.\s
                docName - The document name (must be an image, for example,
book.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
    }
}
```

```
String status = getJobResults(textractClient, jobId);
System.out.println("The job status is " + status);
textractClient.close();
}

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](#) a Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Transcribe usando o SDK para 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 Transcribe.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço específicas, é possível ver as ações contextualizadas em seus devidos cenários e exemplos entre serviços.

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

Listar trabalhos de transcrição

O exemplo de código a seguir mostra como listar trabalhos de transcrição do Amazon Transcribe.

SDK para Java 2.x

Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listTranscriptionJobs(transcribeClient);
    }

    public static void listTranscriptionJobs(TranscribeClient transcribeClient)
    {
```

```

        ListTranscriptionJobsRequest listJobsRequest =
ListTranscriptionJobsRequest.builder()
        .build();

transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
        .flatMap(response -> response.transcriptionJobSummaries().stream())
        .forEach(jobSummary -> {
            System.out.println("Job Name: " +
jobSummary.transcriptionJobName());
            System.out.println("Job Status: " +
jobSummary.transcriptionJobStatus());
            System.out.println("Output Location: " +
jobSummary.outputLocationType());
            // Add more information as needed

            // Retrieve additional details for the job if necessary
            GetTranscriptionJobResponse jobDetails =
transcribeClient.getTranscriptionJob(
                GetTranscriptionJobRequest.builder()
                    .transcriptionJobName(jobSummary.transcriptionJobName())
                    .build());

                // Display additional details
                System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
                System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
                // Add more details as needed

                System.out.println("-----");
            });
        }
    }

```

- Para obter detalhes da API, consulte [ListTranscriptionJobs](#) na Referência AWS SDK for Java 2.x da API.

Iniciar um trabalho de transcrição

O exemplo de código a seguir mostra como iniciar um trabalho de transcrição do Amazon Transcribe.

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }
    }
}
```

```

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString())
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {
                        System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
    }

```

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

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);
```

```
SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
    this.subscriber = s;
    this.inputStream = inputStream;
}

@Override
public void request(long n) {
    if (n <= 0) {
        subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
    }

    demand.getAndAdd(n);

    executor.submit(() -> {
        try {
            do {
                ByteBuffer audioBuffer = getNextEvent();
                if (audioBuffer.remaining() > 0) {
                    AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                    subscriber.onNext(audioEvent);
                } else {
                    subscriber.onComplete();
                    break;
                }
            } while (demand.decrementAndGet() > 0);
        } catch (Exception e) {
            subscriber.onError(e);
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    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();
    }
}
```

- Para obter detalhes da API, consulte [StartTranscriptionJob](#) Referência AWS SDK for Java 2.x da API.

Cenários

Transcrever áudio e obter dados do trabalho

O código de exemplo a seguir mostra como:

- Inicie um trabalho de transcrição com o Amazon Transcribe.
- Aguarde a conclusão do trabalho .
- Obtenha a URI em que a transcrição está armazenada.

Para obter mais informações, consulte [Começar a usar o Amazon Transcribe](#).

SDK para Java 2.x

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWS Code Examples Repository](#).

Transcreve um arquivo PCM.

```
/**
 * To run this AWS code example, ensure that you have set up 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 TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

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

        final String USAGE = "\n" +
            "Usage:\n" +
            "  <file> \n\n" +
            "Where:\n" +
            "  file - the location of a PCM file to transcribe. In this
example, ensure the PCM file is 16 hertz (Hz). \n";

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

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
            .region(REGION)
            .build();
    }
}
```

```
        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromFile(file)),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromFile(String file) {
    try {
        File inputFile = new File(file);
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;

    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US)
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
            System.out.println("Error Occurred: " + sw.toString());
        })
        .onComplete(() -> {
            System.out.println("=== All records stream successfully ===");
        })
        .subscriber(event -> {
```

```
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
            }
        }
    })
    .build();
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
        this.subscriber = s;
    }
}
```

```
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }

    @Override
    public void cancel() {
        executor.shutdown();
    }

    private ByteBuffer getNextEvent() {
        ByteBuffer audioBuffer = null;
        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();
}
}
}

```

Transcreve o streaming de áudio do microfone do computador.

```

public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
        LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

        result.get();
        client.close();
    }
}

```

```
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        })
        .onError(e -> {
            System.out.println(e.getMessage());
            StringWriter sw = new StringWriter();
            e.printStackTrace(new PrintWriter(sw));
        });
}
```

```

        System.out.println("Error Occurred: " + sw.toString());
    })
    .onComplete(() -> {
        System.out.println("=== All records stream successfully ===");
    })
    .subscriber(event -> {
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
                }
            }
        })
    .build();
}

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {

```

```

        this.currentSubscription.cancel();
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    }
    s.onSubscribe(currentSubscription);
}
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
    {
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }
}

```



```
        }
    });
}

@Override
public void cancel() {
    executor.shutdown();
}

private ByteBuffer getNextEvent() {
    ByteBuffer audioBuffer = null;
    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();
}
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API do AWS SDK for Java 2.x .
 - [GetTranscriptionJob](#)
 - [StartTranscriptionJob](#)

Exemplos de serviço cruzado usando o SDK para Java 2.x

Os aplicativos de exemplo a seguir usam o AWS SDK for Java 2.x para trabalhar conjuntamente múltiplos Serviços da AWS.

Os exemplos de serviços cruzados visam um nível avançado de experiência para ajudar a começar a criar aplicações.

Exemplos

- [Criar uma aplicação para enviar dados para uma tabela do DynamoDB](#)
- [Criar um chatbot Amazon Lex para engajar os visitantes do site](#)
- [Criar uma aplicação de publicação e assinatura que traduz mensagens](#)
- [Criar 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](#)
- [Detecte PPE em imagens com o Amazon Rekognition usando um SDK AWS](#)
- [Detecte objetos em imagens com o Amazon Rekognition usando um SDK AWS](#)
- [Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS](#)
- [Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um SDK AWS](#)
- [Usar o API Gateway para invocar uma função do Lambda](#)
- [Usar Step Functions para invocar funções do Lambda](#)
- [Usar eventos programados para chamar 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

Criar um chatbot Amazon Lex para engajar os visitantes do 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 utilizados 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 utilizados neste exemplo

- Amazon SNS

- Amazon Translate

Criar 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 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 utilizados 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 utilizados neste exemplo

- API Gateway
- 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 utilizados 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 utilizados 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 com 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 utilizados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly

- Amazon Textract
- Amazon Translate

Detecte PPE em imagens com o Amazon Rekognition usando um SDK AWS

SDK para Java 2.x

Mostra como criar uma AWS Lambda função 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

Detecte objetos em imagens com o Amazon Rekognition usando um SDK AWS

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 utilizados neste exemplo

- Amazon Rekognition

- Amazon S3
- Amazon SES

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um SDK AWS

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 utilizados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um SDK AWS

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 utilizados 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 AWS Lambda função usando a API de tempo de execução Lambda Java. Este exemplo invoca AWS serviços diferentes para realizar 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 utilizados 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 fluxo de trabalho AWS sem servidor usando AWS Step Functions e. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

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 chamar 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 AWS serviços diferentes para realizar 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 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 você AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como a Segurança da nuvem e a Segurança na nuvem.

Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa todos os serviços oferecidos na AWS nuvem e fornecer serviços que você possa usar com segurança. Nossa responsabilidade de segurança é a maior prioridade em AWS, e a eficácia de nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos [Programas de AWS Conformidade](#).

Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você está usando e por outros fatores, incluindo a sensibilidade de seus dados, os requisitos da sua organização e as leis e regulamentos aplicáveis.

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço, consulte a página de documentação de segurança](#) do serviço e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

Tópicos

- [Proteção de dados em AWS SDK for Java 2.x](#)
- [Trabalhar com TLS no SDK para Java](#)
- [Identity and Access Management](#)
- [Validação de conformidade para este AWS produto ou serviço](#)
- [Resiliência para este AWS produto ou serviço](#)
- [Segurança da infraestrutura para este AWS produto ou serviço](#)

Proteção de dados em AWS SDK for Java 2.x

O modelo de [responsabilidade AWS compartilhada modelo](#) se aplica à proteção de dados em AWS SDK for Java. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura

global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as [Perguntas frequentes sobre privacidade de dados](#). Para mais informações sobre a proteção de dados na Europa, consulte o artigo [AWS Shared Responsibility Model and GDPR](#) no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou 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. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão 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 AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint 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 e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo 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.

Trabalhar com TLS no SDK para 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 ele Serviços 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 compatíveis.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

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, executar um aplicativo Java que use TLS.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

O aplicativo registra o handshake SSL semelhante ao exemplo a seguir.

```
...
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",
    }
  )
...
```

Aplicar uma versão mínima do TLS

O SDK para 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, embora o TLS 1.3 esteja disponível, forneça a seguinte propriedade do sistema.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWS Atualização dos endpoints da API para o 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.

Identity and Access Management

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar AWS os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- [Público](#)
- [Autenticando com identidades](#)

- [Gerenciamento do acesso usando políticas](#)
- [Como Serviços da AWS trabalhar com o IAM](#)
- [Solução de problemas AWS de identidade e acesso](#)

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz AWS.

Usuário do serviço — Se você Serviços da AWS costuma fazer seu trabalho, seu administrador fornece as credenciais e as permissões de que você precisa. À medida que você usa mais AWS recursos para fazer seu trabalho, talvez precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudá-lo a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um recurso no AWS, consulte [Solução de problemas AWS de identidade e acesso](#) o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Administrador de serviços — Se você é responsável pelos AWS recursos da sua empresa, provavelmente tem acesso total AWS a. É seu trabalho determinar quais AWS recursos e recursos seus usuários do serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como sua empresa pode usar o IAM com 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 ver exemplos de políticas AWS baseadas em identidade que você pode usar no IAM, consulte o guia do usuário do AWS service (Serviço da AWS) que você está usando.

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e 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. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte [Como fazer login Conta da AWS](#) no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte [Assinatura de solicitações de AWS API](#) 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, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte [Autenticação multifator](#) no Guia do usuário do AWS IAM Identity Center e [Usar a autenticação multifator \(MFA\) na AWS](#) no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail 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 usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade.

Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte [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 sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. 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 [Altere 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 perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte [Quando criar um usuário do IAM \(em vez de um perfil\)](#) no Guia do usuário do IAM.

Perfis do IAM

Uma [função do IAM](#) é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ela é semelhante a um usuário do IAM, mas não está associada a uma pessoa específica. Você pode assumir temporariamente uma função do IAM no AWS Management Console [trocando de funções](#). Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para o uso de perfis, consulte [Usar perfis do IAM](#) no Guia do usuário do IAM.

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, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. 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 [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 (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Como os perfis 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 da entidade principal da chamada, usando um perfil de serviço ou uma função vinculada ao serviço.
 - **Sessões de acesso direto (FAS)** — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte [Encaminhar sessões de acesso](#).
- **Perfil de serviço:** um perfil de serviço é um [perfil 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 ao serviço** — Uma função vinculada ao 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. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.
- **Aplicativos em execução no Amazon EC2** — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. É preferível fazer isso e armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte [Usar um perfil 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 os perfis do IAM, consulte [Quando criar um perfil do IAM \(em vez de um usuário\)](#) no Guia do usuário do IAM.

Gerenciamento do acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um 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 é armazenada 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 políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissão 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 a 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 função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como um usuário do IAM, grupo de usuários ou função do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte [Criação de política 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 perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas 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 recurso

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 perfil 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 uma entidade principal especificada pode executar nesse recurso e em que condições. Você deve [especificar uma entidade principal](#) em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha que estão localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis 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.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF 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

AWS oferece suporte a tipos de políticas adicionais 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 atributo 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 (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS que sua empresa possui. Se você habilitar todos os atributos 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 nas contas dos membros, incluindo cada uma 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 um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil 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 AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte [Lógica de avaliação de políticas](#) no Guia do usuário do IAM.

Como Serviços da AWS trabalhar com o IAM

Para ter uma visão de alto nível de como Serviços da AWS funciona com a maioria dos recursos do IAM, consulte [AWS os serviços que funcionam com o IAM](#) no Guia do usuário do IAM.

Para saber como usar um específico AWS service (Serviço da AWS) com o IAM, consulte a seção de segurança do Guia do usuário do serviço relevante.

Solução de problemas AWS de identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com AWS um IAM.

Tópicos

- [Não estou autorizado a realizar uma ação em AWS](#)
- [Não estou autorizado a realizar iam: PassRole](#)
- [Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS recursos](#)

Não estou autorizado a realizar uma ação em 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 atributo *my-example-widget* fictício, mas não tem as permissões *aws:GetWidget* fictícias.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

Nesse caso, a política do usuário mateojackson deve ser atualizada para permitir o acesso ao atributo *my-example-widget* usando a ação *aws:GetWidget*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não está autorizado a executar a ação `iam:PassRole`, as suas políticas devem ser atualizadas para permitir que você passe uma função para o AWS.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazê-lo, você deve ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta utilizar 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 precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS recursos

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. 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:

- Para saber se é AWS compatível com esses recursos, consulte [Como Serviços da AWS trabalhar com o IAM](#).

- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como [fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui](#) no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como [fornecer acesso Contas da AWS 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 AWS produto ou serviço

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte [Serviços da AWS Escopo por Programa de Conformidade](#) [Serviços da AWS](#) e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de [AWS conformidade Programas AWS](#) de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte [Baixar relatórios em AWS Artifact](#) .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- [Guias de início rápido sobre segurança e conformidade](#) — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- [Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services](#) — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

Note

Nem todos Serviços da AWS são elegíveis para a HIPAA. Para mais informações, consulte a [Referência dos serviços qualificados pela HIPAA](#).

- AWS Recursos de <https://aws.amazon.com/compliance/resources/> de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- [AWS Guias de conformidade do cliente](#) — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).
- [Avaliação de recursos com regras](#) no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- [AWS Security Hub](#)— Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno 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](#)— Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço, consulte a página de documentação de segurança](#) do serviço e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

Resiliência para este AWS produto ou serviço

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade.

Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância.

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 datacenter tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte [Infraestrutura AWS global](#).

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço](#), consulte a [página de documentação de segurança](#) do serviço e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

Segurança da infraestrutura para este AWS produto ou serviço

Esse AWS produto ou serviço usa serviços gerenciados e, portanto, é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte [AWS Cloud Security](#). Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte [Proteção](#) de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar este AWS Produto ou Serviço pela rede. Os clientes precisam oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com sigilo de encaminhamento perfeito (perfect forward secrecy, ou PFS) como DHE (Ephemeral Diffie-Hellman, ou Efêmero Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman, ou Curva elíptica efêmera Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas utilizando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o [AWS Security Token Service](#) (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Esse AWS produto ou serviço segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para AWS obter informações sobre segurança do [AWS serviço, consulte a página de documentação de segurança](#) do serviço e os [AWS serviços que estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

Migre da versão 1.x para 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 orientações sobre como migrar o código da versão 1.x para a 2.x.

Tópicos

- [Novidades da versão 2](#)
- [step-by-step Instruções de migração com exemplo](#)
- [O que há de diferente entre o AWS SDK for Java 1.x e o 2.x](#)
- [Use o SDK para Java 1.x e 2.x lado a lado](#)

Novidades da versão 2

- Você também pode configurar seus próprios clientes HTTP. Consulte [Configuração de transporte HTTP](#).
- Os clientes assíncronos oferecem suporte de E/S sem bloqueio e objetos de retorno. `CompletableFuture` Consulte [Programação assíncrona](#).
- As operações que retornam várias páginas possuem respostas autopaginadas. Isso permite que você concentre o código no que vai ser feito com a resposta, sem a necessidade de verificar e obter páginas subsequentes. Consulte [Paginação](#).
- O desempenho das AWS Lambda funções no horário de início do SDK foi aprimorado. Consulte [Melhorias de desempenho do horário de início do SDK](#).
- A versão 2.x oferece 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 para exemplos de códigos específicos, consulte o as outras seções neste guia.

- [Início rápido](#)

- [Configuração](#)
- [Exemplos de código para o AWS SDK for Java 2.x](#)
- [Usar o SDK](#)
- [Segurança para o AWS SDK for Java](#)

step-by-step Instruções de migração com exemplo

Esta seção fornece um step-by-step guia para migrar seu aplicativo que atualmente usa o SDK for Java v1.x para o SDK for Java 2.x. A primeira parte apresenta uma visão geral das etapas seguidas por um exemplo detalhado de uma migração.

As etapas abordadas aqui descrevem a migração de um caso de uso normal, em que o aplicativo chama Serviços da AWS usando clientes de serviço orientados por modelos. Se você precisar migrar código que usa APIs de nível superior, como o [S3 Transfer Manager](#) ou a [CloudFront pré-assinatura](#), consulte a seção abaixo [the section called “O que é diferente entre o 1.x e o 2.x”](#) do índice.

A abordagem descrita aqui é uma sugestão. Você pode usar outras técnicas e aproveitar os recursos de edição de código do seu IDE para alcançar o mesmo resultado.

Visão geral das etapas

1. Comece adicionando o SDK for Java 2.x BOM

Ao adicionar o elemento Maven BOM (Lista de materiais) do SDK for Java 2.x ao seu arquivo POM, você garante que todas as dependências v2 necessárias sejam da mesma versão. Seu POM pode conter dependências v1 e v2. Isso permite que você migre seu código de forma incremental em vez de alterá-lo de uma só vez.

SDK para Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
```

```
</dependencies>  
</dependencyManagement>
```

Você pode encontrar a [versão mais recente](#) no Repositório Central do Maven.

2. Pesquisar arquivos para instruções de importação da classe v1

Ao escanear os arquivos em seu aplicativo para importações v1, você encontrará dependências v2 para adicionar ao seu arquivo Maven POM.

3. Determine as dependências do Maven v2 a partir das instruções de importação da v1

Depois de encontrar todas as instruções de importação v1 exclusivas, você pode determinar o artefato Maven correspondente para a dependência v2 consultando o nome do pacote para a tabela de mapeamento de dependências.

4. Adicione elementos de dependência v2 ao arquivo POM

Atualize o arquivo POM do Maven com os elementos de dependência determinados na etapa 3.

5. Nos arquivos Java, altere incrementalmente as classes v1 para as classes v2

Ao fazer isso, faça as alterações necessárias para dar suporte à API v2, como usar construtores em vez de construtores e usar getters e setters fluentes.

6. Remova as dependências v1 do Maven das importações POM e v1 dos arquivos

Ao fazer isso, faça as alterações necessárias para dar suporte à API v2, como usar construtores em vez de construtores e usar os getters e setters fluentes.

7. Refatore o código para usar os aprimoramentos da API v2

Depois que o código for compilado com sucesso e passar nos testes, você poderá aproveitar os aprimoramentos da v2, como usar um cliente HTTP ou paginadores diferentes para simplificar o código. Esta é uma etapa opcional.

Exemplo de migração

Neste exemplo, migramos um aplicativo que usa o SDK for Java v1 e acessa vários. Serviços da AWS Trabalhamos detalhadamente com o seguinte método v1 na etapa 5. Esse é um método em uma classe que contém oito métodos e há 32 classes no aplicativo.

método v1 para migrar

Somente as importações do SDK v1 estão listadas abaixo no arquivo Java.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple requests.
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.

            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.

                    if (RUNNING_STATES.contains(instance.getState().getName())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.getNextToken() != null);
    } catch (final AmazonEC2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
    }
}
```

```

        if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}

```

1. Adicionar V2 Maven BOM

Adicione a BOM do Maven para o SDK for Java 2.x ao POM junto com quaisquer outras dependências na seção. `dependencyManagement` Se seu arquivo POM tiver a BOM para v1 do SDK, deixe-a por enquanto. Ele será removido em uma etapa posterior.

Gerenciamento de dependências do POM desde o início

```

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>                <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>    <!--Existing v1 BOM dependency. -->
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>software.amazon.awssdk</groupId>    <!--Add v2 BOM dependency. -->
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

```



```
</dependencyManagement>
```

2. Pesquisar arquivos para instruções de importação da classe v1

Pesquise o código do aplicativo em busca de ocorrências exclusivas de `import com.amazonaws.services`. Isso nos ajuda a determinar as dependências v1 usadas pelo projeto. Se seu aplicativo tiver um arquivo Maven POM com dependências v1 listadas, você poderá usar essas informações em vez disso.

Neste exemplo, usamos o comando [ripgrep\(rg\)](#) para pesquisar a base de código.

Da raiz da sua base de código, execute o `ripgrep` comando a seguir. Depois de `ripgrep` encontrar as instruções de importação, elas são canalizadas para os `uniq` comandos `cut` e `sort`, e para isolar os nomes dos serviços.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

Para esse aplicativo, o seguinte é registrado no console.

```
autoscaling
cloudformation
ec2
identitymanagement
```

Isso indica que houve pelo menos uma ocorrência de cada um dos seguintes nomes de pacotes usados nas `import` declarações. Quatro de nossos propósitos, os nomes das classes individuais não importam. Só precisamos encontrar os serviços que são usados.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

3. Determine as dependências do Maven v2 a partir das instruções de importação da v1

Os nomes de serviço para v1 que isolamos da Etapa 2 — por exemplo `autoscaling` e `cloudformation` — podem ser mapeados para o mesmo nome de serviço v2 em sua maior parte.

Como o artifactID v2 Maven corresponde ao nome do serviço na maioria dos casos, você tem as informações necessárias para adicionar blocos de dependência ao seu arquivo POM.

A tabela a seguir mostra como podemos determinar as dependências da v2.

O nome do serviço v1 é mapeado para... nome do pacote	O nome do serviço v2 é mapeado para... nome do pacote	Dependência do Maven v2
ec2 com.amazonaws.services. ec2 .*	ec2 software.amazon.awssdk.services. ec2 .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> ec2</ artifactId> </dependency></pre>
escalonamento automático com.amazonaws.services. autoscaling .*	escalonamento automático software.amazon.awssdk.services. autoscaling .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> autoscali ng </artifactId> </dependency></pre>
cloudformation com.amazonaws.services. cloudform ation .*	cloudformation software.amazon.awssdk. cloudform ation .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> cloudform ation </artifactId> </dependency></pre>
gerenciamento de identidade* com.amazonaws.services. identitym anagement .*	eu sou* software.amazon.awssdk. iam .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> iam</ artifactId></pre>

O nome do serviço v1 é mapeado para...	O nome do serviço v2 é mapeado para...	Dependência do Maven v2
nome do pacote	nome do pacote	<code></dependency></code>

* O iam mapeamento `identitymanagement` para é uma exceção em que o nome do serviço usado no nome do pacote difere entre as versões.

4. Adicione elementos de dependência v2 ao arquivo POM

Na etapa 3, determinamos os quatro blocos de dependência que precisam ser adicionados ao arquivo POM. Não precisamos adicionar uma versão porque especificamos o BOM na etapa 1. Depois que as importações são adicionadas, nosso arquivo POM tem os seguintes elementos de dependência.

```

...
<dependencies>
  ...
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>autoscaling</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cloudformation</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>ec2</artifactId>
  </dependency>
  ...
</dependencies>
...

```

5. Nos arquivos Java, altere incrementalmente as classes v1 para as classes v2

No método que estamos migrando, vemos

- Um cliente de serviço EC2 da `com.amazonaws.services.ec2.AmazonEC2Client`.
- Várias classes de modelo EC2 usadas. Por exemplo `DescribeInstancesRequest` e `DescribeInstancesResult`.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesRequest;
import com.amazonaws.services.ec2.model.DescribeInstancesResult;
import com.amazonaws.services.ec2.model.Instance;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Reservation;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;
...
private static List<Instance> getRunningInstances(AmazonEC2Client ec2, List<String>
instanceIds)
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = new DescribeInstancesRequest()
            .withInstanceIds(instanceIds);
        DescribeInstancesResult result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens with
multiple re
            result = ec2.describeInstances(request);
            request.setNextToken(result.getNextToken()); // Prepare request for next
page.
            for (final Reservation r : result.getReservations()) {
                for (final Instance instance : r.getInstances()) {
                    LOGGER.info("Examining instanceId: "+ instance.getInstanceId());
                    // if instance is in a running state, add it to runningInstances
list.
                    if (RUNNING_STATES.contains(instance.getState().getName())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.isTruncated());
    } catch (AmazonEC2Exception e) {
        // ...
    }
}
```

```
        }
    }
} while (result.getNextToken() != null);
} catch (final AmazonEC2Exception exception) {
    // if instance isn't found, assume its terminated and continue.
    if (exception.getErrorCode().equals(NOT_FOUND_ERROR_CODE)) {
        LOGGER.info("Instance probably terminated; moving on.");
    } else {
        throw exception;
    }
}
return runningInstances;
}
...

```

Nosso objetivo é substituir todas as importações v1 por importações v2. Prosseguimos uma aula por vez.

a. Substitua a instrução de importação ou o nome da classe

Vemos que o primeiro parâmetro do `describeRunningInstances` método é uma `AmazonEC2Client` instância v1. Execute um destes procedimentos:

- Substitua a importação com `amazonaws.services.ec2.AmazonEC2Client` por `software.amazon.awssdk.services.ec2.Ec2Client` e altere `AmazonEC2Client` para `Ec2Client`.
- Altere o tipo de parâmetro para `Ec2Client` e deixe que o IDE solicite a importação correta. Nosso IDE solicitará que importemos a classe v2 porque os nomes dos clientes são diferentes — `AmazonEC2Client` e `Ec2Client`. Essa abordagem não funciona se o nome da classe for o mesmo nas duas versões.

b. Substitua as classes do modelo v1 por equivalentes v2

Após a mudança para a `v2Ec2Client`, se usarmos um IDE, veremos erros de compilação na declaração a seguir.

```
result = ec2.describeInstances(request);
```

O erro de compilação resulta do uso de uma instância de v1 `DescribeInstancesRequest` como parâmetro para o método `Ec2Client describeInstances` v2. Para corrigir, faça as seguintes declarações de substituição ou importação.

replace	with
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. Mude os construtores v1 para construtores v2.

Ainda vemos erros de compilação porque [não há construtores nas classes v2](#). Para corrigir, faça a seguinte alteração.

alteração	para
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. Substitua objetos de ***Result** resposta v1 por equivalentes ***Response** v2

Uma diferença consistente entre v1 e v2 é que todos os [objetos de resposta na v2 terminam com *Response em](#) vez de. `*Result` Substitua a `DescribeInstancesResult` importação v1 pela importação v2, `DescribeInstancesResponse`

d. Faça alterações na API

A declaração a seguir precisa de algumas mudanças.

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

Na v2, os [métodos setter](#) não usam o set ou com. prefix Os métodos Getter get prefixados com também foram incluídos no SDK for Java 2.x

Classes de modelo, como a request instância, são imutáveis na v2, então precisamos criar uma nova DescribeInstancesRequest com um construtor.

Na v2, a declaração se torna a seguinte.

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.nextToken())
    .build();
```

d. Repita até que o método seja compilado com classes v2

Continue com o resto do código. Substitua as importações v1 pelas importações v2 e corrija os erros de compilação. Consulte a referência da [API v2 e a referência *What's different*](#), conforme necessário.

Depois de migrarmos esse único método, temos o seguinte código v2.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.RegionUtils;
import com.amazonaws.services.ec2.AmazonEC2Client;
import com.amazonaws.services.ec2.model.AmazonEC2Exception;
import com.amazonaws.services.ec2.model.CreateTagsRequest;
import com.amazonaws.services.ec2.model.InstanceStateName;
import com.amazonaws.services.ec2.model.Tag;
import com.amazonaws.services.ec2.model.TerminateInstancesRequest;

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.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.Reservation;
...
private static List<Instance> getRunningInstances(Ec2Client ec2, List<String>
instanceIds) {
    List<Instance> runningInstances = new ArrayList<>();
    try {
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(instanceIds)
            .build();
```

```

        DescribeInstancesResponse result;
        do {
            // DescribeInstancesResponse is a paginated response, so use tokens
with multiple re
            result = ec2.describeInstances(request);
            request = DescribeInstancesRequest.builder() // Prepare request for
next page.
                .nextToken(result.nextToken())
                .build();
            for (final Reservation r : result.reservations()) {
                for (final Instance instance : r.instances()) {
                    // if instance is in a running state, add it to
runningInstances list.
                    if (RUNNING_STATES.contains(instance.state().nameAsString())) {
                        runningInstances.add(instance);
                    }
                }
            }
        } while (result.nextToken() != null);
    } catch (final Ec2Exception exception) {
        // if instance isn't found, assume its terminated and continue.
        if (exception.awsErrorDetails().errorCode().equals(NOT_FOUND_ERROR_CODE)) {
            LOGGER.info("Instance probably terminated; moving on.");
        } else {
            throw exception;
        }
    }
    return runningInstances;
}
...

```

Como estamos migrando um único método em um arquivo Java com oito métodos, temos uma combinação de importações v1 e v2 à medida que trabalhamos no arquivo. Adicionamos as últimas seis instruções de importação à medida que executamos as etapas.

Depois de migrarmos todo o código, não haverá mais instruções de importação v1.

6. Remova as dependências v1 do Maven das importações POM e v1 dos arquivos

Depois de migrarmos todo o código v1 no arquivo, temos as seguintes instruções de importação do SDK v2.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
```



```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.ServiceMetadata;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Instance;
import software.amazon.awssdk.services.ec2.model.InstanceStateName;
import software.amazon.awssdk.services.ec2.model.Reservation;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
```

Depois de migrarmos todos os arquivos em nosso aplicativo, não precisaremos mais das dependências v1 em nosso arquivo POM. Remova o BOM v1 da seção `DependencyManagement`, se estiver usando, e todos os blocos de dependência v1.

7. Refatore o código para usar os aprimoramentos da API v2

Para o trecho que estamos migrando, podemos usar opcionalmente um paginador v2 e deixar o SDK gerenciar as solicitações baseadas em tokens para obter mais dados.

Podemos substituir toda a do cláusula pela seguinte.

```
DescribeInstancesIterable responses =
ec2.describeInstancesPaginator(request);

responses.reservations().stream()
    .forEach(reservation -> reservation.instances()
        .forEach(instance -> {
            if
(RUNNING_STATES.contains(instance.state().nameAsString())) {
                runningInstances.put(instance.instanceId(),
instance);
            }
        }));
```

O que há de diferente entre o AWS SDK for Java 1.x e o 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 alteração notável do SDK para Java 1.x para o SDK para 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 para o SDK 2.x. Os artefatos do Maven para o SDK 2.x usam o groupId `software.amazon.awssdk`, enquanto o SDK 1.x usa o groupId `com.amazonaws`.

Algumas vezes, seu código exige uma dependência de `com.amazonaws` para um projeto que, de outra forma, usa somente artefatos do SDK 2.x. Um exemplo disso é quando você trabalha com o AWS Lambda do lado do servidor. Isso foi mostrado na seção [Configurar um projeto do Apache Maven](#), anteriormente neste guia.

Note

Vários nomes de pacotes no SDK 1.x contêm a v2. O uso da 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:

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

Adicionar 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.x 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>
```

Você também pode [usar as versões 1.x e 2.x side-by-side](#) ao migrar seu projeto para a versão 2.x.

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.

Exemplo de atualizar um objeto de solicitação na 1.x

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

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

Exemplo 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();
```

Métodos setter e getter

Na AWS SDK for Java versão 2.x, os nomes dos métodos setter não incluem o prefixo `set` ou `with`. Por exemplo, `*.withEndpoint()` é `*.endpoint()` agora.

Os nomes dos métodos Getter não usam o `get` prefixo.

Exemplo de usar métodos setter em 1.x

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

Exemplo de usar métodos setter em 2.x

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

Exemplo de usar métodos getter em 1.x

```
String token = request.getNextToken();
```

Exemplo de usar métodos getter em 2.x

```
String token = request.nextToken();
```

Nomes de classes de modelo

Os nomes das classes de modelo que representam as respostas do serviço terminam `Response` em v2 em vez dos `Result` que a v1 usa.

Exemplo de nomes de classes que representam uma resposta na v1

```
CreateApiKeyResult
AllocateAddressResult
```

Exemplo de nomes de classes que representam uma resposta na v2

```
CreateApiKeyResponse
```

AllocateAddressResponse

Status da migração de bibliotecas e utilitários

Bibliotecas e utilitários do SDK para Java

A tabela a seguir lista o status da migração de bibliotecas e utilitários para o 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
CloudFrontUrlSigner, CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3 TransferManager	2.19.0
Cliente de metadados do EC2	Cliente de metadados do EC2	2.19.29
Analizador de URI do S3	Analizador de URI do S3	2.20.41
Criador de política do IAM	Criador de política do IAM	2.20.126
Buffer do lado do cliente do Amazon SQS	Processamento em lotes 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 para Java 2.x.

Nome usado com a versão 2.x do SDK para Java	Desde a versão
Cliente de criptografia do Amazon S3	3.0.0 1

Nome usado com a versão 2.x do SDK para Java	Desde a versão
AWS Cliente 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 dependência do Maven a seguir.

```
<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>
```

Detalhes da migração para bibliotecas e utilitários

- [Gerenciador de transferências S3](#)
- [Utilitário de metadados do EC2](#)
- [CloudFront pré-assinando](#)
- [Análise de URI do S3](#)

Alterações de cliente

Builders do cliente

Você deve criar todos os clientes usando o método do compilador de clientes. Construtores não estão mais disponíveis.

Exemplo de criar um cliente na versão 1.x

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

Exemplo de criar um cliente na versão 2.x

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

Nomes de classes de clientes

Todos os nomes de classe de cliente agora são totalmente concatenados e não são mais armazenados com o prefixo Amazon. Estas alterações estão alinhadas com os nomes usados no AWS CLI.

Exemplo dos nomes de classe na 1.x

```
AmazonDynamoDB
AWSACMPAAsyncClient
```

Exemplo dos nomes de classe na 2.x

```
DynamoDbClient
AcmAsyncClient
```

Alterações no nome da classe do cliente

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.acmpca.AWSACMPAAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.acmpca.AWSACMPClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexforbusiness.AlexaForBusinessAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.alex aforbusiness.AmazonAlexaFor BusinessClient</code>	<code>software.amazon.awssdk.serv ices.alexaforbusiness.Alexa ForBusinessClient</code>
<code>com.amazonaws.services.apig ateway.AmazonApiGatewayAsyn cClient</code>	<code>software.amazon.awssdk.serv ices.apigateway.ApiGatewayA syncClient</code>
<code>com.amazonaws.services.apig ateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.serv ices.apigateway.ApiGatewayC lient</code>
<code>com.amazonaws.services.appl icationautoscaling.AWSAppli cationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.serv ices.applicationautoscaling .ApplicationAutoScalingAsyn cClient</code>
<code>com.amazonaws.services.appl icationautoscaling.AWSAppli cationAutoScalingClient</code>	<code>software.amazon.awssdk.serv ices.applicationautoscaling .ApplicationAutoScalingClient</code>
<code>com.amazonaws.services.appl icationdiscovery.AWSAppl icationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.serv ices.applicationdiscovery.A pplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.appl icationdiscovery.AWSAppl icationDiscoveryClient</code>	<code>software.amazon.awssdk.serv ices.applicationdiscovery.A pplicationDiscoveryClient</code>
<code>com.amazonaws.services.app stream.AmazonAppStreamAsynC lient</code>	<code>software.amazon.awssdk.serv ices.appstream.AppStreamAsy ncClient</code>
<code>com.amazonaws.services.app stream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.serv ices.appstream.AppStreamClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.appsync.AWSAppSyncAsyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncAsyncClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaAsyncClient</code>	<code>software.amazon.awssdk.services.athena.AthenaAsyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaClient</code>	<code>software.amazon.awssdk.services.athena.AthenaClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient</code>
<code>com.amazonaws.services.batch.AWSBatchAsyncClient</code>	<code>software.amazon.awssdk.services.batch.BatchAsyncClient</code>
<code>com.amazonaws.services.batch.AWSBatchClient</code>	<code>software.amazon.awssdk.services.batch.BatchClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsAsyncClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.budgets.AWSBudgetsClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9AsyncClient</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9AsyncClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9Client</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9Client</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.codecommit.AWSCodeCommitClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarAsyncClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarAsyncClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendAsyncClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendAsyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendClient</code>
<code>com.amazonaws.services.config.AmazonConfigAsyncClient</code>	<code>software.amazon.awssdk.services.config.ConfigAsyncClient</code>
<code>com.amazonaws.services.config.AmazonConfigClient</code>	<code>software.amazon.awssdk.services.config.ConfigClient</code>
<code>com.amazonaws.services.connect.AmazonConnectAsyncClient</code>	<code>software.amazon.awssdk.services.connect.ConnectAsyncClient</code>
<code>com.amazonaws.services.connect.AmazonConnectClient</code>	<code>software.amazon.awssdk.services.connect.ConnectClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineAsyncClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.dax.AmazonDaxAsyncClient</code>	<code>software.amazon.awssdk.services.dax.DaxAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.dax. AmazonDaxClient</code>	<code>software.amazon.awssdk.serv ices.dax.DaxClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmA syncClient</code>
<code>com.amazonaws.services.devi cefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.serv ices.devicefarm.DeviceFarmC lient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctAsyncClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectAsyncClient</code>
<code>com.amazonaws.services.dire ctconnect.AmazonDirectConne ctClient</code>	<code>software.amazon.awssdk.serv ices.directconnect.DirectCo nnectClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceAs yncClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryAsy ncClient</code>
<code>com.amazonaws.services.dire ctory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.serv ices.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMAsyncClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm. AmazonDLMClient</code>	<code>software.amazon.awssdk.serv ices.dlm.DlmClient</code>
<code>com.amazonaws.services.dyna modbv2.AmazonDynamoDBAsynC lient</code>	<code>software.amazon.awssdk.serv ices.dynamodb.DynamoDbAsynC Client</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2AsyncClient</code>	<code>software.amazon.awssdk.services.ec2.Ec2AsyncClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2Client</code>	<code>software.amazon.awssdk.services.ec2.Ec2Client</code>
<code>com.amazonaws.services.ecr.AmazonECRAsyncClient</code>	<code>software.amazon.awssdk.services.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr.AmazonECRClient</code>	<code>software.amazon.awssdk.services.ecr.EcrClient</code>
<code>com.amazonaws.services.ecs.AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.services.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs.AmazonECSClient</code>	<code>software.amazon.awssdk.services.ecs.EcsClient</code>
<code>com.amazonaws.services.eks.AmazonEKSAsyncClient</code>	<code>software.amazon.awssdk.services.eks.EksAsyncClient</code>
<code>com.amazonaws.services.eks.AmazonEKSClient</code>	<code>software.amazon.awssdk.services.eks.EksClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.elasticache.AsyncClient</code>	<code>software.amazon.awssdk.services.elasticache.ElasticCacheAsyncClient</code>
<code>com.amazonaws.services.elasticache.ElasticCacheClient</code>	<code>software.amazon.awssdk.services.elasticache.ElasticCacheClient</code>
<code>com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkAsyncClient</code>	<code>software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkAsyncClient</code>
<code>com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkClient</code>	<code>software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkClient</code>
<code>com.amazonaws.services.elasticfilesystem.AsyncClient</code>	<code>software.amazon.awssdk.services.efs.EfsAsyncClient</code>
<code>com.amazonaws.services.elasticfilesystem.ElasticFileSystemClient</code>	<code>software.amazon.awssdk.services.efs.EfsClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancing.ElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient</code>	<code>software.amazon.awssdk.services.emr.EmrAsyncClient</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient</code>	<code>software.amazon.awssdk.services.emr.EmrClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient</code>
<code>com.amazonaws.services.fms.AWSFMSAsyncClient</code>	<code>software.amazon.awssdk.services.fms.FmsAsyncClient</code>
<code>com.amazonaws.services.fms.AWSFMSClient</code>	<code>software.amazon.awssdk.services.fms.FmsClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.gamelift.AmazonGameLiftClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierAsyncClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierAsyncClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierClient</code>
<code>com.amazonaws.services.glue.AWSGlueAsyncClient</code>	<code>software.amazon.awssdk.services.glue.GlueAsyncClient</code>
<code>com.amazonaws.services.glue.AWSGlueClient</code>	<code>software.amazon.awssdk.services.glue.GlueClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassAsyncClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassAsyncClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyClient</code>
<code>com.amazonaws.services.health.AWSHealthAsyncClient</code>	<code>software.amazon.awssdk.services.health.HealthAsyncClient</code>
<code>com.amazonaws.services.health.AWSHealthClient</code>	<code>software.amazon.awssdk.services.health.HealthClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient</code>	<code>software.amazon.awssdk.services.iam.IamAsyncClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient</code>	<code>software.amazon.awssdk.services.iam.IamClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportAsyncClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportAsyncClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportClient</code>	<code>software.amazon.awssdk.services.importexport.ImportExportClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorAsyncClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorAsyncClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorClient</code>
<code>com.amazonaws.services.iot.AWSIoTAsyncClient</code>	<code>software.amazon.awssdk.services.iot.IotAsyncClient</code>
<code>com.amazonaws.services.iot.AWSIoTClient</code>	<code>software.amazon.awssdk.services.iot.IotClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataAsyncClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataAsyncClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisAsyncClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisAsyncClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseAsyncClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient</code>	Sem suporte
<code>com.amazonaws.services.kms.AWSKMSAsyncClient</code>	<code>software.amazon.awssdk.services.kms.KmsAsyncClient</code>
<code>com.amazonaws.services.kms.AWSKMSClient</code>	<code>software.amazon.awssdk.services.kms.KmsClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaAsyncClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaAsyncClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailAsyncClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailAsyncClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailClient</code>
<code>com.amazonaws.services.logs.AWSLogsAsyncClient</code>	<code>software.amazon.awssdk.services.logs.LogsAsyncClient</code>
<code>com.amazonaws.services.logs.AWSLogsClient</code>	<code>software.amazon.awssdk.services.logs.LogsClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningClient</code>
<code>com.amazonaws.services.macie.AmazonMacieAsyncClient</code>	<code>software.amazon.awssdk.services.macie.MacieAsyncClient</code>
<code>com.amazonaws.services.macie.AmazonMacieClient</code>	<code>software.amazon.awssdk.services.macie.MacieClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveAsyncClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveAsyncClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient</code>
<code>com.amazonaws.services.mediatailor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediatailor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.migrationhub.AWSMigrationHubClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubClient</code>
<code>com.amazonaws.services.mobile.AWSMobileAsyncClient</code>	<code>software.amazon.awssdk.services.mobile.MobileAsyncClient</code>
<code>com.amazonaws.services.mobile.AWSMobileClient</code>	<code>software.amazon.awssdk.services.mobile.MobileClient</code>
<code>com.amazonaws.services.mq.AmazonMQAsyncClient</code>	<code>software.amazon.awssdk.services.mq.MqAsyncClient</code>
<code>com.amazonaws.services.mq.AmazonMQClient</code>	<code>software.amazon.awssdk.services.mq.MqClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkAsyncClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkAsyncClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneAsyncClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneAsyncClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.opsworks.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OperationsWorksCMAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworks.OperationsWorksCMClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsAsyncClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsAsyncClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsClient</code>
<code>com.amazonaws.services.pi.AWSPIAsyncClient</code>	<code>software.amazon.awssdk.services.pi.PiAsyncClient</code>
<code>com.amazonaws.services.pi.AWSPIClient</code>	<code>software.amazon.awssdk.services.pi.PiClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointAsyncClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointClient</code>
<code>com.amazonaws.services.polly.AmazonPollyAsyncClient</code>	<code>software.amazon.awssdk.services.polly.PollyAsyncClient</code>
<code>com.amazonaws.services.polly.AmazonPollyClient</code>	<code>software.amazon.awssdk.services.polly.PollyClient</code>
<code>com.amazonaws.services.pricing.AWS PricingAsyncClient</code>	<code>software.amazon.awssdk.services.pricing.PricingAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.pricing.AWS PricingClient</code>	<code>software.amazon.awssdk.services.pricing.PricingClient</code>
<code>com.amazonaws.services.rds.AmazonRDSAsyncClient</code>	<code>software.amazon.awssdk.services.rds.RdsAsyncClient</code>
<code>com.amazonaws.services.rds.AmazonRDSClient</code>	<code>software.amazon.awssdk.services.rds.RdsClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftAsyncClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftAsyncClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionAsyncClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53AsyncClient</code>	<code>software.amazon.awssdk.services.route53.Route53AsyncClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53Client</code>	<code>software.amazon.awssdk.services.route53.Route53Client</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsClient</code>
<code>com.amazonaws.services.s3.AmazonS3Client</code>	<code>software.amazon.awssdk.services.s3.S3Client</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient</code>	<code>software.amazon.awssdk.services.sts.StsAsyncClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient</code>	<code>software.amazon.awssdk.services.sts.StsClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient</code>	<code>software.amazon.awssdk.services.sms.SmsAsyncClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationClient</code>	<code>software.amazon.awssdk.services.sms.SmsClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient</code>
<code>com.amazonaws.services.shield.AWSShieldAsyncClient</code>	<code>software.amazon.awssdk.services.shield.ShieldAsyncClient</code>
<code>com.amazonaws.services.shield.AWSShieldClient</code>	<code>software.amazon.awssdk.services.shield.ShieldClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbAsyncClient</code>
<code>com.amazonaws.services.simplesdb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.services.simplesdb.SimpleDbClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient</code>	<code>software.amazon.awssdk.services.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient</code>	<code>software.amazon.awssdk.services.ses.SesClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.simp lesystemsmanagement.AWSSimp leSystemsManagementAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ssm.SsmAsyncClient</code>
<code>com.amazonaws.services.simp lesystemsmanagement.AWSSimp leSystemsManagementClient</code>	<code>software.amazon.awssdk.serv ices.ssm.SsmClient</code>
<code>com.amazonaws.services.simp leworkflow.AmazonSimpleWork flowAsyncClient</code>	<code>software.amazon.awssdk.serv ices.swf.SwfAsyncClient</code>
<code>com.amazonaws.services.simp leworkflow.AmazonSimpleWork flowClient</code>	<code>software.amazon.awssdk.serv ices.swf.SwfClient</code>
<code>com.amazonaws.services.snow ball.AmazonSnowballAsyncClient</code>	<code>software.amazon.awssdk.serv ices.snowball.SnowballAsyn cClient</code>
<code>com.amazonaws.services.snow ball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.serv ices.snowball.SnowballClient</code>
<code>com.amazonaws.services.sns. AmazonSNSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsAsyncClient</code>
<code>com.amazonaws.services.sns. AmazonSNSClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsAsyncClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.stepfunctions.AWSStepFunctionsAsyncClient</code>	<code>software.amazon.awssdk.services.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.stepfunctions.AWSStepFunctionsClient</code>	<code>software.amazon.awssdk.services.sfn.SfnClient</code>
<code>com.amazonaws.services.storagegateway.AWSStorageGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.storagegateway.StorageGatewayAsyncClient</code>
<code>com.amazonaws.services.storagegateway.AWSStorageGatewayClient</code>	<code>software.amazon.awssdk.services.storagegateway.StorageGatewayClient</code>
<code>com.amazonaws.services.support.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.services.support.SupportAsyncClient</code>
<code>com.amazonaws.services.support.AWSSupportClient</code>	<code>software.amazon.awssdk.services.support.SupportClient</code>
<code>com.amazonaws.services.transcribe.AmazonTranscribeAsyncClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeAsyncClient</code>
<code>com.amazonaws.services.transcribe.AmazonTranscribeClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateAsyncClient</code>	<code>software.amazon.awssdk.services.translate.TranslateAsyncClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateClient</code>	<code>software.amazon.awssdk.services.translate.TranslateClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.waf.AWSWAFAsyncClient</code>	<code>software.amazon.awssdk.services.waf.WafAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFClient</code>	<code>software.amazon.awssdk.services.waf.WafClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalAsyncClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailAsyncClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailAsyncClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkSpacesClient</code>

Cliente 1.x	Cliente 2.x
<code>com.amazonaws.services.xray.AWSXRayAsyncClient</code>	<code>software.amazon.awssdk.services.xray.XRayAsyncClient</code>
<code>com.amazonaws.services.xray.AWSXRayClient</code>	<code>software.amazon.awssdk.services.xray.XRayClient</code>

Padrões de criação de clientes

Na versão 2.x, as seguintes alterações foram feitas na lógica padrão de criação do cliente.

- A cadeia de provedores de credenciais padrão para o S3 não inclui mais credenciais anônimas. Você deve especificar manualmente o acesso anônimo ao S3 usando o `AnonymousCredentialsProvider`
- As seguintes variáveis de ambiente relacionadas à criação padrão do cliente são diferentes.

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 propriedades do sistema a seguir relacionadas à criação padrão do cliente são diferentes.

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>

- A versão 2.x não suporta as seguintes propriedades do sistema.

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 carga de região de um arquivo `endpoints.json` personalizado não tem mais suporte.

Configuração do cliente

Na 1.x, a configuração do cliente do SDK foi modificada pela configuração de uma instância de `ClientConfiguration` no cliente ou no builder do cliente. Na versão 2.x, a configuração do cliente é dividida em classes de configuração separadas. As classes de configuração separadas permitem configurar diferentes clientes HTTP para clientes assíncronos versus síncronos, mas ainda usam a mesma classe `ClientOverrideConfiguration`.

Exemplo da configuração do cliente na versão 1.x

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

Exemplo 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();
```

Exemplo 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();
```

Cientes HTTP

Mudanças notáveis

- Na versão 2.x, você pode alterar qual cliente HTTP usar em tempo de execução especificando uma implementação usando `clientBuilder.httpClientBuilder`
- Quando você passa um cliente HTTP usando `clientBuilder.httpClient` para um construtor de clientes de serviço, o cliente HTTP não é fechado por padrão se o cliente de serviço fechar. Isso permite que você compartilhe clientes HTTP entre clientes de serviço.
- Clientes HTTP assíncronos agora usam E/S sem bloqueio.
- Algumas operações agora usam HTTP/2 para melhorar o desempenho.

Alterações nas configurações

Configuração	1.x	Sincronização 2.x, Apache	2.x Assíncrono, Netty
	<pre>ClientCon figuration clientConfig = new ClientCon figuration()</pre>	<pre>ApacheHtt pClient.B uilder httpClien tBuilder = ApacheHtt pClient.b uilder()</pre>	<pre>NettyNioA syncHttpC lient.Builder httpClien tBuilder = NettyNioA syncHttpC lient.builder()</pre>
Conexões máximas	<pre>clientCon fig.setMa xConnecti ons(...) clientCon fig.withM axConnect ions(...)</pre>	<pre>httpClien tBuilder. maxConnec tions(...)</pre>	<pre>httpClien tBuilder. maxConcur rency(...)</pre>
Tempo limite da conexão	<pre>clientCon fig.setCo nnectionT imeout(...) clientConfig.wi thConnect ionTimeout(...)</pre>	<pre>httpClien tBuilder. connectio nTimeout(...)</pre>	<pre>httpClien tBuilder. connectio nTimeout(...)</pre>
Tempo limite do soquete	<pre>clientCon fig.setSo cketTimeo ut(...) clientConf ig.withSo cketTimeo ut(...)</pre>	<pre>httpClien tBuilder. socketTim eout(...)</pre>	<pre>httpClien tBuilder. writeTime out(...) httpClien tBuilder. readTimeo ut(...)</pre>

Configuração	1.x	Sincronização 2.x, Apache	2.x Assíncrono, Netty
Conexão TTL	<pre>clientConfig.setConnectionTTL(...) clientConfig.withConnectionTTL(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>	<pre>httpClientBuilder.connectionTimeToLive(...)</pre>
Conexão máxima em marcha lenta	<pre>clientConfig.setConnectionMaxIdleMillis(...) clientConfig.withConnectionMaxIdleMillis(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>	<pre>httpClientBuilder.connectionMaxIdleTime(...)</pre>
Validar após inatividade	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	Não suportado (recurso de solicitação)	Não suportado (recurso de solicitação)

Configuração	1.x	Sincronização 2.x, Apache	2.x Assíncrono, Netty
Endereço local	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	Sem suporte
Esperar continuar ativado	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	Não suportado (recurso de solicitação)
Cefeiro de conexões	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDBClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>	<pre>DynamoDBAsyncClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

Proxies de cliente HTTP

Configurações	1.x	Sincronização 2.x, Apache	2.x Assíncrono, Netty
	<pre>ClientCon figuration clientConfig = new ClientCon figuration()</pre>	<pre>ProxyConf iguration .Builder proxyConfig = ProxyConf iguration .builder()</pre>	<pre>ProxyConf iguration .Builder proxyConfig = ProxyConf iguration .builder()</pre>
Host do proxy	<pre>clientCon fig.setPr oxyHost(...) clientConfig.w ithProxyH ost(...)</pre>	<pre>proxyConf ig.endpoi nt(...)</pre>	<pre>proxyConf ig.host(...)</pre>
Porta do proxy	<pre>clientCon fig.setPr oxyPort(...) clientConfig.w ithProxyP ort(...)</pre>	<pre>proxyConf ig.endpoi nt(...)</pre> <p>A porta proxy está incorporada em endpoint</p>	<pre>proxyConf ig.port(...)</pre>
Nome de usuário do proxy	<pre>clientCon fig.setPr oxyUserna me(...) clientConf ig.withPr oxyUserna me(...)</pre>	<pre>proxyConf ig.userna me(...)</pre>	<pre>proxyConf ig.userna me(...)</pre>

Configurações	1.x	Sincronização 2.x, Apache	2.x Assíncrono, Netty
Senha do proxy	<pre>clientConfig.setProxyPassword(...) clientConfig.withProxyPassword(...)</pre>	<pre>proxyConfig.setPassword(...)</pre>	<pre>proxyConfig.setPassword(...)</pre>
Domínio proxy	<pre>clientConfig.setProxyDomain(...) clientConfig.withProxyDomain(...)</pre>	<pre>proxyConfig.setNtlmDomain(...)</pre>	Não suportado (recurso de solicitação)
Estação de trabalho proxy	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.setNtlmWorkstation(...)</pre>	Não suportado (recurso de solicitação)
Métodos de autenticação de proxy	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	Não suportado	Não suportado (recurso de solicitação)

Configurações	1.x	Sincronização 2.x, Apache	2.x Assíncrono, Netty
Autenticação preemptiva básica de proxy	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	Não suportado (recurso de solicitação)
Hosts sem proxy	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>
Desativar proxy de soquete	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	Não suportado (recurso de solicitação)	Não suportado (recurso de solicitação)

Configurações	1.x	Sincronização 2.x, Apache	2.x Assíncrono, Netty
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>httpClientBuilder. proxyConfiguration(proxyConfiguration) .build()</pre>	<pre>httpClientBuilder. proxyConfiguration(proxyConfiguration) .build()</pre>

Substituições do cliente

Configuração	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ClientOverrideConfiguration.Builder overrideConfig = ClientOverrideConfiguration.builder()</pre>
Prefixo do agente do usuário	<pre>clientConfig.setUserAgentPrefix(...) clientConfig.withUserAgentPrefix(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)</pre>
Sufixo do agente do usuário	<pre>clientConfig.setUserAgentSuffix(...) clientConfig.withUserAgentSuffix(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)</pre>

Configuração	1.x	2.x
Signer	<pre>clientConfig.setSignerOverride(...) clientConfig.withSignerOverride(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.SIGNER, ...)</pre>
Cabeçalhos adicionais	<pre>clientConfig.addHeader(...) clientConfig.withHeader(...)</pre>	<pre>overrideConfig.putHeader(...)</pre>
Tempo limite da solicitação	<pre>clientConfig.setRequestTimeout(...) clientConfig.withRequestTimeout(...)</pre>	<pre>overrideConfig.apiCallAttemptTimeout(...)</pre>
Tempo limite de execução do cliente	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
Use o Gzip	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	Não suportado (recurso de solicitação)
Dica de tamanho do buffer de soquete	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	Não suportado (recurso de solicitação)

Configuração	1.x	2.x
Metadados de resposta em cache	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	Não suportado (recurso de solicitação)
Tamanho do cache de metadados de resposta	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	Não suportado (recurso de solicitação)
resolvedor de DNS	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	Não suportado (recurso de solicitação)
TCP mantém-se vivo	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	Essa opção agora está na configuração do cliente HTTP <ul style="list-style-type: none"> - <code>ApacheHttpClient.builder().tcpKeepAlive(true)</code> - <code>NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</code>
Seguro aleatório	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	Não suportado (recurso de solicitação)

Configuração	1.x	2.x
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

O cliente anula novas tentativas

Configuração	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>RetryPolicy.Builder retryPolicy = RetryPolicy.builder()</pre>
Erro máximo de repetição	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
Use novas tentativas limitadas	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	Sem suporte
Máximo de tentativas consecutivas antes da limitação	<pre>clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...) clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)</pre>	Sem suporte

Configuração	1.x	2.x
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

Cientes assíncronos

Configuração	1.x	2.x
		<pre>ClientAsyncConfiguration.Builder asyncConfig = ClientAsyncConfiguration.builder()</pre>
Executor	<pre>AmazonDynamoDBAsyncClientBuilder.standard() .withExecutorFactory(...) .build()</pre>	<pre>asyncConfig.advancedOption(SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)</pre>
		<pre>DynamoDbAsyncClient.builder() .asyncConfiguration(asyncConfig) .build()</pre>

Outras mudanças no cliente

A `ClientConfiguration` opção a seguir, da 1.x, foi alterada na versão 2.x do SDK e não tem um equivalente direto.

Configuração	1.x	Equivalente a 2.x
Protocolo	<pre>clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)</pre>	<p>A configuração do protocolo é HTTPS por padrão. Para modificar a configuração, especifique a configuração do protocolo de um endpoint HTTP no construtor do cliente:</p> <pre>clientBuilder.endpointOverride(URI.create("http://..."))</pre>

Alterações no provedor de credenciais

Esta seção fornece um mapeamento das alterações de nomes de classes e métodos do provedor de credenciais entre as versões 1.x e 2.x do AWS SDK for Java.

Diferenças notáveis

- O provedor de credenciais padrão carrega as propriedades do sistema antes das variáveis de ambiente na versão 2.x. Para obter mais informações, consulte [Uso de 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 [Configuração da AWS CLI](#) no Guia do usuário da AWS Command Line Interface.

Alterações no provedor de credenciais mapeadas entre as versões 1.x e 2.x

AWSCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.AWSCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.AwsCredentialsProvider</code>
Nome do método	<code>getCredentials</code>	<code>resolveCredentials</code>
Método não suportado	<code>refresh</code>	Não suportado

DefaultAWSCredentialsProviderChain

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code>

Alterar categoria	1.x	2.x
Criação	<code>new DefaultAWSCredentialsProviderChain</code>	<code>DefaultCredentialsProvider.create</code>
Método não suportado	<code>getInstance</code>	Não suportado
Ordem de prioridade das configurações externas	Variáveis de ambiente antes das propriedades do sistema	Propriedades do sistema antes das variáveis de ambiente

AWSStaticCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
Criação	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>

EnvironmentVariableCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.EnvironmentVariableCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider</code>

Alterar categoria	1.x	2.x
Criação	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>
Nome da variável de ambiente	<code>AWS_ACCESS_KEY</code>	<code>AWS_ACCESS_KEY_ID</code>
	<code>AWS_SECRET_KEY</code>	<code>AWS_SECRET_ACCESS_KEY</code>

SystemPropertiesCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.SystemPropertiesCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.SystemPropertiesCredentialsProvider</code>
Criação	<code>new SystemPropertiesCredentialsProvider</code>	<code>SystemPropertiesCredentialsProvider.create</code>
Nome da propriedade do sistema	<code>aws.secretKey</code>	<code>aws.secretAccessKey</code>

ProfileCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.profile.ProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider</code>

Alterar categoria	1.x	2.x
Criação	<code>new ProfileCredentialsProvider</code>	<code>ProfileCredentialsProvider.create</code>
Localização do perfil personalizado	<ul style="list-style-type: none"> • <code>AWS_CREDENTIAL_FILES_FILE</code> variável de ambiente • <code>new ProfileCredentialsProvider</code> 	<ul style="list-style-type: none"> • <code>AWS_SHARED_CREDENTIALS_FILE</code> variável de ambiente • <code>ProfileCredentialsProvider.builder</code>

ContainerCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.ContainerCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code>
Criação	<code>new ContainerCredentialsProvider</code>	<code>ContainerCredentialsProvider.create</code>
Especificar atualização assíncrona	Não suportado	Comportamento padrão

InstanceProfileCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>

Alterar categoria	1.x	2.x
Criação	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
Especificar atualização assíncrona	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialsProvider.builder().asyncCredentialUpdateEnabled(true).build()</code>
Nome da propriedade do sistema	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

STSAssumeRoleSessionCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>
Criação	<ul style="list-style-type: none"> <code>new STSAssumeRoleSessionCredentialsProvider</code> <code>new STSAssumeRoleSessionCredentialsProvider.Builder</code> 	<code>StsAssumeRoleCredentialsProvider.builder</code>
Atualização assíncrona	Comportamento padrão	Comportamento padrão

Alterar categoria	1.x	2.x
Configuração	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	Configurar uma <code>AssumeRoleRequest</code> solicitação <code>StsClient</code> e

STSSessionCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
Criação	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Atualização assíncrona	Comportamento padrão	<code>StsGetSessionTokenCredentialsProvider.builder</code>
Configuração	Parâmetros do construtor	Configurar uma <code>GetSessionTokenRequest</code> solicitação <code>StsClient</code> e em um construtor

WebIdentityFederationSessionCredentialsProvider

Alterar categoria	1.x	2.x
Nome do pacote/classe	<code>com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.WebIdentityFederationSessionCredentialsProvider</code>

Alterar categoria	1.x	2.x
	<code>SessionCredentialsProvider</code>	<code>auth.StsAssumeRoleWithWebIdentityCredentialsProvider</code>
Criação	<code>new WebIdentityFederationSessionCredentialsProvider</code>	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
Atualização assíncrona	Comportamento padrão	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
Configuração	Parâmetros do construtor	Configurar uma <code>AssumeRoleWithWebIdentityRequest</code> solicitação <code>StsClient</code> e em um construtor

Classes substituídas

Classe 1.x	Classes de substituição 2.x
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> e <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> e <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

Classes removidas

Classe 1.x

```
com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider
```

```
com.amazonaws.auth.PropertiesFileCredentialsProvider
```

Mudanças na região

Esta seção descreve as mudanças implementadas no AWS SDK for Java 2.x para usar as classes `Region` e `Regions`.

Configuração de região

- Alguns serviços da AWS 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 nome de método e classe

As seguintes tabelas mapeiam 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);
```

1.x Alterações no método da classe `Regions`

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>

1.x	2.x
<code>Regions.getName</code>	<code>Region.id</code>
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	Sem suporte
<code>Regions.DEFAULT_REGION</code>	Sem suporte
<code>Regions.name</code>	<code>Region.id</code>

1.x Alterações no método da classe de região

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>
<code>Region.hasHttpsEndpoint</code>	Sem suporte
<code>Region.hasHttpEndpoint</code>	Sem suporte
<code>Region.getAvailableEndpoints</code>	Sem suporte
<code>Region.createClient</code>	Sem suporte

RegionMetadata mudanças no método de classe

1.x	2.x
<code>RegionMetadata.getName</code>	<code>RegionMetadata.name</code>
<code>RegionMetadata.getDomain</code>	<code>RegionMetadata.domain</code>
<code>RegionMetadata.getPartition</code>	<code>RegionMetadata.partition</code>

ServiceMetadata mudanças no método de classe

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

Alterações nas operações, solicitações e respostas

Na versão 2.x do SDK for Java, as solicitações são passadas para uma operação do cliente. Por exemplo, `DynamoDbClient's PutItemRequest` é passado para `DynamoDbClient.putItem` operação. Essas operações retornam uma resposta do AWS service (Serviço da AWS), como `PutItemResponse`.

A versão 2.x do SDK for Java tem as seguintes alterações em relação à 1.x.

- As operações com várias páginas de resposta agora têm um `Paginator` método para iterar automaticamente todos os itens na resposta.
- Você não pode alterar solicitações e respostas.
- Você deve criar solicitações e respostas com um método construtor estático em vez de um construtor. Por exemplo, `1.x new PutItemRequest().withTableName(...)` é agora `PutItemRequest.builder().tableName(...).build()`.
- As operações oferecem suporte a uma forma abreviada de criar solicitações: `dynamoDbClient.putItem(request -> request.tableName(...))`.

Operações de streaming

Operações de streaming, como Amazon S3 `getObject` e `putObject` métodos, agora oferecem suporte a E/S sem bloqueio. Como resultado, os POJOs de solicitação e resposta não usam mais um como parâmetro. `InputStream` Em vez disso, para solicitações síncronas, o objeto de solicitação aceita `RequestBody`, o que é um fluxo de bytes. O equivalente assíncrono aceita um `AsyncRequestBody`.

Exemplo da operação **putObject** do Amazon S3 na versão 1.x

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

Exemplo da operação **putObject** do Amazon S3 na versão 2.x

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

Em paralelo, um objeto de resposta de streaming aceita um `ResponseTransformer` para clientes síncronos e um `AsyncResponseTransformer` para clientes assíncronos.

Exemplo da operação **getObject** do Amazon S3 na versão 1.x

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

Exemplo da operação **getObject** do Amazon S3 na versão 2.x

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

No SDK for Java 2.x, as operações de resposta de streaming têm `AsBytes` um método para carregar a resposta na memória e simplificar as conversões de tipo comuns na memória.

Alterações na exceção

Os nomes das classes de exceção, suas estruturas e seus relacionamentos foram alterados. `software.amazon.awssdk.core.exception.SdkException` é a nova `Exception` classe base que todas as outras exceções estendem.

Esta tabela mapeia as alterações no nome da classe de exceção.

1.x	2.x
<code>com.amazonaws.SdkBaseException</code> <code>com.amazonaws.AmazonClientException</code>	<code>software.amazon.awssdk.core.exception.SdkException</code>
<code>com.amazonaws.SdkClientException</code>	<code>software.amazon.awssdk.core.exception.SdkClientException</code>
<code>com.amazonaws.AmazonServiceException</code>	<code>software.amazon.awssdk.awscore.exception.AwsServiceException</code>

A tabela a seguir mapeia os métodos em classes de exceção entre a versão 1.x e a 2.x.

1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>
<code>AmazonServiceException.getErrorCode</code>	<code>AwsServiceException.awsErrorDetails().errorCode</code>
<code>AmazonServiceException.getErrorMessage</code>	<code>AwsServiceException.awsErrorDetails().errorMessage</code>
<code>AmazonServiceException.getStatusCode</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode</code>
<code>AmazonServiceException.getHttpHeaders</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().headers</code>

1.x	2.x
<code>AmazonServiceException.rawResponse</code>	<code>AwsServiceException.awsErrorDetails().rawResponse</code>

Alterações na serialização

O SDK para Java v1.x e v2.x diferem na forma como serializam objetos `List` para solicitar parâmetros.

O SDK para Java 1.x não serializa uma lista vazia, enquanto o SDK para Java 2.x serializa uma lista vazia como um parâmetro vazio.

Por exemplo, pense em um serviço com uma `SampleOperation` que exija uma `SampleRequest`. A `SampleRequest` aceita dois parâmetros: um tipo de `string` `str1` e um tipo de lista `listParam`: conforme mostrado nos exemplos a seguir.

Exemplo de **SampleOperation** na 1.x

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

O registro em log em nível de transferência mostra que o parâmetro `listParam` não está serializado.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Exemplo de **SampleOperation** na 2.x

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

O registro em log em nível de transferência mostra que o parâmetro `listParam` está serializado sem valor.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```


Alterações específicas do serviço

Mudanças no Amazon S3

O SDK for Java 2.x desativa o acesso anônimo por padrão. Como resultado, você deve habilitar o acesso anônimo usando `AnonymousCredentialsProvider` o.

Alterações no nome da operação

Muitos dos nomes de operação para o cliente do Amazon S3 foram alterados no AWS SDK for Java 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.

A tabela a seguir mostra os nomes das operações nas duas versões.

Nomes de operação do Amazon S3

1.x	2.x
<code>abortMultipartUpload</code>	<code>abortMultipartUpload</code>
<code>changeObjectStorageClass</code>	<code>copyObject</code>
<code>completeMultipartUpload</code>	<code>completeMultipartUpload</code>
<code>copyObject</code>	<code>copyObject</code>
<code>copyPart</code>	<code>uploadPartCopy</code>
<code>createBucket</code>	<code>createBucket</code>
<code>deleteBucket</code>	<code>deleteBucket</code>
<code>deleteBucketAnalyticsConfiguration</code>	<code>deleteBucketAnalyticsConfiguration</code>
<code>deleteBucketCrossOriginConfiguration</code>	<code>deleteBucketCors</code>
<code>deleteBucketEncryption</code>	<code>deleteBucketEncryption</code>

1.x	2.x
deleteBucketInventoryConfiguration	deleteBucketInventoryConfiguration
deleteBucketLifecycleConfiguration	deleteBucketLifecycle
deleteBucketMetricsConfiguration	deleteBucketMetricsConfiguration
deleteBucketPolicy	deleteBucketPolicy
deleteBucketReplicationConfiguration	deleteBucketReplication
deleteBucketTaggingConfiguration	deleteBucketTagging
deleteBucketWebsiteConfiguration	deleteBucketWebsite
deleteObject	deleteObject
deleteObjectTagging	deleteObjectTagging
deleteObjects	deleteObjects
deleteVersion	deleteObject
disableRequesterPays	putBucketRequestPayment
doesBucketExist	headBucket
doesBucketExistV2	headBucket
doesObjectExist	headObject
enableRequesterPays	putBucketRequestPayment
generatePresignedUrl	S3Presigner
getBucketAccelerateConfiguration	getBucketAccelerateConfiguration

1.x	2.x
<code>getBucketAcl</code>	<code>getBucketAcl</code>
<code>getBucketAnalyticsConfiguration</code>	<code>getBucketAnalyticsConfiguration</code>
<code>getBucketCrossOriginConfiguration</code>	<code>getBucketCors</code>
<code>getBucketEncryption</code>	<code>getBucketEncryption</code>
<code>getBucketInventoryConfiguration</code>	<code>getBucketInventoryConfiguration</code>
<code>getBucketLifecycleConfiguration</code>	<code>getBucketLifecycle</code> ou <code>getBucketLifecycleConfiguration</code>
<code>getBucketLocation</code>	<code>getBucketLocation</code>
<code>getBucketLoggingConfiguration</code>	<code>getBucketLogging</code>
<code>getBucketMetricsConfiguration</code>	<code>getBucketMetricsConfiguration</code>
<code>getBucketNotificationConfiguration</code>	<code>getBucketNotification</code> ou <code>getBucketNotificationConfiguration</code>
<code>getBucketPolicy</code>	<code>getBucketPolicy</code>
<code>getBucketReplicationConfiguration</code>	<code>getBucketReplication</code>
<code>getBucketTaggingConfiguration</code>	<code>getBucketTagging</code>
<code>getBucketVersioningConfiguration</code>	<code>getBucketVersioning</code>
<code>getBucketWebsiteConfiguration</code>	<code>getBucketWebsite</code>
<code>getObject</code>	<code>getObject</code>
<code>getObjectAcl</code>	<code>getObjectAcl</code>
<code>getObjectAsString</code>	<code>getObjectAsBytes().asUtf8String</code>

1.x	2.x
getObjectMetadata	headObject
getObjectTagging	getObjectTagging
getResourceUrl	S3Utilities#getUrl
getS3AccountOwner	listBuckets
getUrl	S3Utilities#getUrl
headBucket	headBucket
initiateMultipartUpload	createMultipartUpload
isRequesterPaysEnabled	getBucketRequestPayment
listBucketAnalyticsConfigurations	listBucketAnalyticsConfigurations
listBucketInventoryConfigurations	listBucketInventoryConfigurations
listBucketMetricsConfigurations	listBucketMetricsConfigurations
listBuckets	listBuckets
listMultipartUploads	listMultipartUploads
listNextBatchOfObjects	listObjectsV2Paginator
listNextBatchOfVersions	listObjectVersionsPaginator
listObjects	listObjects
listObjectsV2	listObjectsV2
listParts	listParts
listVersions	listObjectVersions

1.x	2.x
putObject	putObject
restoreObject	restoreObject
restoreObjectV2	restoreObject
selectObjectContent	selectObjectContent
setBucketAccelerateConfiguration	putBucketAccelerateConfiguration
setBucketAcl	putBucketAcl
setBucketAnalyticsConfiguration	putBucketAnalyticsConfiguration
setBucketCrossOriginConfiguration	putBucketCors
setBucketEncryption	putBucketEncryption
setBucketInventoryConfiguration	putBucketInventoryConfiguration
setBucketLifecycleConfiguration	putBucketLifecycle ou putBucketLifecycleConfiguration
setBucketLoggingConfiguration	putBucketLogging
setBucketMetricsConfiguration	putBucketMetricsConfiguration
setBucketNotificationConfiguration	putBucketNotification ou putBucketNotificationConfiguration
setBucketPolicy	putBucketPolicy
setBucketReplicationConfiguration	putBucketReplication
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning

1.x	2.x
<code>setBucketWebsiteConfiguration</code>	<code>putBucketWebsite</code>
<code>setObjectAcl</code>	<code>putObjectAcl</code>
<code>setObjectRedirectLocation</code>	<code>copyObject</code>
<code>setObjectTagging</code>	<code>putObjectTagging</code>
<code>uploadPart</code>	<code>uploadPart</code>

Mudanças no Amazon SNS

Um cliente SNS não pode mais acessar tópicos do SNS em regiões diferentes da região para a qual ele está configurado para acessar.

Alterações no Amazon SQS

Um cliente SQS não pode mais acessar filas SQS em regiões diferentes da região que ele está configurado para acessar.

Mudanças no Amazon RDS

O SDK for Java 2.x é `RdsUtilities#generateAuthenticationToken` usado no lugar da `RdsIamAuthTokenGenerator` classe em 1.x.

Alterações no arquivo de perfil

Ele AWS SDK for Java 2.x analisa as definições do perfil em `~/.aws/config` e `~/.aws/credentials` para emular mais de perto a forma como a AWS CLI analisa os arquivos.

O SDK para Java 2.x:

- Resolve um `~/` ou `~` seguido pelo separador de caminho padrão do sistema de arquivos no início do caminho, verificando, em ordem, `$USERPROFILE` (somente Windows), `$HOME`, `$HOMEPATH` (somente Windows) e `$HOMEDRIVE`, em seguida, a propriedade do `user.home` sistema.
- Procura a variável de `AWS_SHARED_CREDENTIALS_FILE` ambiente em vez de `AWS_CREDENTIAL_PROFILES_FILE`.

- Descarta silenciosamente as definições de perfil nos arquivos de configuração sem a palavra `profile` no início do nome do perfil.
- Elimina silenciosamente as definições de perfil que não consistem em caracteres alfanuméricos, sublinhados ou traços (após a `profile` palavra inicial ter sido removida dos arquivos de configuração).
- Mescla as configurações das definições de perfil duplicadas no mesmo arquivo.
- Mescla as configurações das definições de perfil duplicadas nos arquivos de configuração e credenciais.
- NÃO mescla as configurações se ambas `[profile foo]` `[foo]` forem encontradas no mesmo arquivo.
- Usa as configurações `[profile foo]` se ambas `[profile foo]` `[foo]` forem encontradas no arquivo de configuração.
- Usa o valor da última configuração duplicada no mesmo arquivo e perfil.
- Reconhece `;` e `#` por definir um comentário.
- Reconhece `;` e `#` nas definições de perfil define um comentário, mesmo que os caracteres estejam adjacentes ao colchete de fechamento.
- Reconhece `;` e define um comentário somente `#` ao definir valores somente se eles forem precedidos por espaços em branco.
- Reconhece `;` `#` e todo o conteúdo a seguir ao definir valores se eles não forem precedidos por espaços em branco.
- Considera as credenciais baseadas em funções as credenciais de maior prioridade. O SDK 2.x sempre usa credenciais baseadas em funções se o usuário especificar a propriedade. `role_arn`
- Considera as credenciais baseadas em sessão as credenciais. `second-highest-priority` O SDK 2.x sempre usa credenciais baseadas em sessão se as credenciais baseadas em função não forem usadas e o usuário especificar as propriedades `e. aws_access_key_id` `aws_session_token`
- Usa credenciais básicas se as credenciais baseadas em função e sessão não forem usadas e o usuário especificar a propriedade. `aws_access_key_id`

Variáveis de ambiente e alterações nas propriedades do sistema

1.x Variável de ambiente	1.x Propriedade do sistema	Variável de ambiente 2.x	2.x Propriedade do sistema
AWS_ACCESS_KEY_ID AWS_ACCESS_KEY	aws.accessKeyId	AWS_ACCESS_KEY_ID	aws.accessKeyId
AWS_SECRET_KEY AWS_SECRET_ACCESS_KEY	aws.secretKey	AWS_SECRET_ACCESS_KEY	aws.secretAccessKey
AWS_SESSION_TOKEN	aws.sessionToken	AWS_SESSION_TOKEN	aws.sessionToken
AWS_REGION	aws.region	AWS_REGION	aws.region
AWS_CONFIG_FILE		AWS_CONFIG_FILE	aws.configFile
AWS_CREDENTIALS_PROFILE_FILE		AWS_SHARED_CREDENTIALS_FILE	aws.sharedCredentialsFile
AWS_PROFILE	aws.profile	AWS_PROFILE	aws.profile
AWS_EC2_METADATA_DISABLED	com.amazonaws.sdk.disableEc2Metadata	AWS_EC2_METADATA_DISABLED	aws.disableEc2Metadata
	com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	AWS_EC2_METADATA_SERVICE_ENDPOINT	aws.ec2MetadataServiceEndpoint

1.x Variável de ambiente	1.x Propriedade do sistema	Variável de ambiente 2.x	2.x Propriedade do sistema
AWS_CONTA_INER_CREDENTIALS_RELATIVE_URI		AWS_CONTA_INER_CREDENTIALS_RELATIVE_URI	aws.contaInerCredentialsPath
AWS_CONTA_INER_CREDENTIALS_FULL_URI		AWS_CONTA_INER_CREDENTIALS_FULL_URI	aws.contaInerCredentialsFullUri
AWS_CONTA_INER_AUTHORIZATION_TOKEN		AWS_CONTA_INER_AUTHORIZATION_TOKEN	aws.contaInerAuthorizationToken
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIonBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertificateChecking	Não suportado (recurso de solicitação)	Não suportado (recurso de solicitação)

1.x Variável de ambiente	1.x Propriedade do sistema	Variável de ambiente 2.x	2.x Propriedade do sistema
	<code>com.amazonaws.sdk.enableDefaultMetrics</code>	Sem suporte	Sem suporte
	<code>com.amazonaws.sdk.enableThrottledRetry</code>	Sem suporte	Sem suporte
	<code>com.amazonaws.regions.RegionUtils.fileOverride</code>	Não suportado (recurso de solicitação)	Não suportado (recurso de solicitação)
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	Não suportado (recurso de solicitação)	Não suportado (recurso de solicitação)
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	Não suportado (recurso de solicitação)	Não suportado (recurso de solicitação)
	<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>	Não suportado (recurso de solicitação)	Não suportado (recurso de solicitação)

Alterações nos Waiters da versão 1 para a versão 2

Este tópico detalha as alterações na funcionalidade dos Waiters da versão 1 (v1) para a versão 2 (v2).

As tabelas a seguir demonstram a diferença especificamente para garçons do DynamoDB. Os garçons de outros serviços seguem o mesmo padrão.

Alterações de alto nível

As aulas de garçons usam o mesmo artefato Maven do serviço.

Alteração	v1	v2
Dependências do Maven	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.680¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>dynamodb</artif actId> </dependency> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.25.10²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>dynamodb</artif actId> </dependency> </pre>

Alteração	v1	v2
	<code></dependencies></code>	<code></dependencies></code>
Nome do pacote	<code>com.amazonaws.services.dynamodbv2.waiters</code>	<code>software.amazon.awssdk.services.dynamodb.waiters</code>
Nomes da classe	AmazonDynamoDBWaiters	<ul style="list-style-type: none"> Síncrono: DynamoDbWaiter Assíncrono: DynamoDbAsyncWaiter

¹ [Versão mais recente.](#) ² [Versão mais recente.](#)

Mudanças na API

Alteração	v1	v2
Crie um garçom	<pre>AmazonDynamoDB client = AmazonDynamoDBClientBuilder .standard().build(); AmazonDynamoDBWaiters waiter = client.waiters();</pre>	<p>Síncrono:</p> <pre>DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();</pre> <p>Assíncrono:</p> <pre>DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();</pre>

Alteração	v1	v2
Esperare até que exista uma tabela	<p>Síncrono:</p> <pre>waiter.tableExists() .run(new WaiterParameters<>(new DescribeTableRequest(tableName)))</pre> <p>Assíncrono:</p> <pre>waiter.tableExists() .runAsync(new WaiterParameters() .withRequest(new DescribeTableRequest(tableName)), new WaiterHandler() { @Override public void onSuccess(AmazonWebServiceRequest amazonWebServiceRequest) { System.out.println("Table creation succeeded"); } @Override public void onFailure(Exception e) { e.printStackTrace(); } })</pre>	<p>Síncrono:</p> <pre>WaiterResponse<DescribeTableResponse> waiterResponse = waiter.waitUntilTableExists(r -> r.tableName("myTable")); waiterResponse.matched().response() .ifPresent(System.out::println);</pre> <p>Assíncrono:</p> <pre>waiter.waitUntilTableExists(r -> r.tableName(tableName)) .whenComplete((r, t) -> { if (t != null) { t.printStackTrace(); } else { System.out.println("Table creation succeeded"); } }).join();</pre>

Alteração	v1	v2
	<code>}).get();</code>	

Alterações de configuração

Alteração	v1	v2
Estratégia de votação (máximo de tentativas e atraso fixo)	<pre> MaxAttemptsRetryStrategy maxAttemptsRetryStrategy = new MaxAttemptsRetryStrategy(10); FixedDelayStrategy fixedDelayStrategy = new FixedDelayStrategy(3); PollingStrategy pollingStrategy = new PollingStrategy(maxAttemptsRetryStrategy, fixedDelayStrategy); waiter.tableExists().run(new WaiterParameters<>(new DescribeTableRequest(tableName), pollingStrategy); </pre>	<pre> FixedDelayBackoffStrategy fixedDelayBackoffStrategy = FixedDelayBackoffStrategy.create(Duration.ofSeconds(3)); waiter.waitUntilTableExists(r -> r.tableName(tableName), c -> c.maxAttempts(10) .backoffStrategy(fixedDelayBackoffStrategy)); </pre>

Alterações do Gerenciador de transferência do Amazon S3 da versão 1 para a versão 2

Este tópico detalha as alterações no Gerenciador de transferência do Amazon S3 da versão 1 (v1) para a versão 2 (v2).

Alterações de alto nível

Alteração	v1	v2
Dependências do Maven	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587<sup>1</sup></version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-s3</ artifactId> </dependency> </dependencies> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21<sup>2</sup></version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifactId>s3- transfer-manager</art ifactId> </dependency> <dependency> <groupId> software.amazon.aw ssdk.crt</groupId> </pre>

Alteração	v1	v2
		<pre><artifact Id>aws-crt</artifa ctId> <version> 0.28.7³</version> </dependency> </dependencies></pre>
Nome do pacote	com.amazonaws.serv ices.s3.transfer	software.amazon.aw ssdk.transfer.s3
Nome da classe	TransferManager	S3TransferManager

¹ [Versão mais recente](#). ² [Versão mais recente](#). ³ [Versão mais recente](#).

Alterações da API de configuração

Configuração	v1	v2
(obtenha um construtor)	<pre>TransferManagerBuilder tmBuilder = TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder = S3TransferManager. builder();</pre>
Cliente do S3	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
Executor	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
Encerrar grupos de threads	<pre>tmBuilder.withShut DownThreadPools(...);</pre>	Sem suporte. O executor fornecido não será desligado

Configuração	v1	v2
	<pre>tmBuilder.setS hutdownThreadPools (...);</pre>	quando o S3 TransferManager for fechado
Tamanho mínimo da parte de upload	<pre>tmBuilder.withMini mumUploadPartSize(...); tmBuilder.setMinimumU ploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtB uilder(). minimumPa rtSizeInBytes(...) .build(); tmBuilder.s3Clie nt(s3);</pre>
Limite de multipart upload	<pre>tmBuilder.withMini mumUploadPartSize(...); tmBuilder.setMinimumU ploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtB uilder(). threshol dInBytes(...).build(); tmBuilder.s3Client(s3) ;</pre>
Tamanho mínimo da parte de cópia	<pre>tmBuilder.withMini mumUploadPartSize(...); tmBuilder.setMinimumU ploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtB uilder(). minimumPa rtSizeInBytes(...) .build(); tmBuilder.s3Clie nt(s3);</pre>

Configuração	v1	v2
Limite de cópia de multipart	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder(). threshold InBytes(...).build(); tmBuilder.s3Client(s3) ;</pre>
Desabilitar downloads paralelos	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>Desabilite os downloads paralelos passando um cliente S3 padrão baseado em Java para o gerenciador de transferência.</p> <pre>S3AsyncClient s3 = S3AsyncClient.builder().build(); tmBuilder.s3Client(s3);</pre>
Sempre calcule o multipart md5	<pre>tmBuilder.withAlwaysCalculateMultipartMd5(...); tmBuilder.setAlwaysCalculateMultipartMd5(...);</pre>	Sem suporte.

Alteração de comportamento

A transferência paralela requer um cliente AWS S3 baseado em CRT





No SDK para Java 2.x, o recurso de transferência paralela automática (multipart upload/download) está disponível por meio do [cliente do S3 baseado em AWS CRT](#). Para ativar o recurso de



transferência paralela, você deve adicionar explicitamente a dependência da [biblioteca de Common Runtime \(CRT - Tempo de execução comum\) da AWS](#) para maximizar o desempenho.

O cliente S3 AWS baseado em CRT sozinho, sem usar, fornece desempenho maximizado de transferências `S3TransferManager` paralelas. `S3TransferManagerA v2` fornece APIs adicionais que facilitam a transferência de arquivos e diretórios.

A capacidade de `S3TransferManager` realizar transferências paralelas depende de como ela `S3TransferManager` é iniciada e se a biblioteca AWS Common Runtime (CRT) foi declarada como uma dependência.

A tabela a seguir descreve três cenários de inicialização para uma `S3TransferManager v2` com e sem a AWS CRT declarada como dependência.

Abordagem de inicialização do S3 TransferManager v2	O AWS CRT é declarado como uma dependência?	
	sim	não
<p>Inicializar o S3TransferManager sem transmitir uma instância S3AsyncClient</p> <p>Método de criação estático:</p> <pre>S3TransferManager.create();</pre> <p>- OU -</p> <p>Método do criador:</p> <pre>S3TransferManager.builder().build();</pre>	 <p>transferência paralela automática habilitada</p>	 <p>transferência paralela automática desabilitada</p>
<p>Transmitir uma instância S3AsyncClient criada com um dos métodos do criador <code>crt*</code>()</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	 <p>transferência paralela</p>	 <p>erro de runtime</p>

Abordagem de inicialização do S3 TransferManager v2	O AWS CRT é declarado como uma dependência?	
<p>- OU -</p> <pre data-bbox="115 331 1019 531">S3AsyncClient s3AsyncClient = S3AsyncClient.crtC reate(); S3TransferManager.builder().s3AsyncClient(s3A syncClient).build();</pre>	<p>automátic a habilitada</p>	
<p>Transmitir uma instância S3AsyncClient criada com um dos métodos padrão do criador para que o gerente de transferência não tenha nenhuma referência ao CRT</p> <pre data-bbox="115 772 1019 972">S3AsyncClient s3AsyncClient = S3AsyncClient.buil der().build(); S3TransferManager.builder().s3AsyncClient(s3Asyn cClient).build();</pre> <p>- OU -</p> <pre data-bbox="115 1077 1019 1276">S3AsyncClient s3AsyncClient = S3AsyncClient.crea te(); S3TransferManager.builder().s3AsyncClient(s3Asyn cClient).build();</pre>	<div data-bbox="1094 604 1222 716" style="text-align: center;"></div> <p>transferê ncia paralela automática desabilitada</p>	<div data-bbox="1334 604 1463 716" style="text-align: center;"></div> <p>transferê ncia paralela automática desabilitada</p>

Download paralelo por meio de buscas de intervalo de bytes

Quando o recurso de transferência paralela automática está ativado, o Gerenciador de transferência do S3 v2 usa [buscas de intervalo de bytes](#) para recuperar partes específicas do objeto em paralelo (download multipart). A forma como um objeto é baixado com a v2 não depende de como o objeto foi originalmente carregado. Todos os downloads podem se beneficiar da alta taxa de transferência e da simultaneidade.

Por outro lado, com o Gerenciador de transferência do S3 v1, faz diferença como o objeto foi originalmente carregado. O Gerenciador de transferência do S3 v1 recupera as partes do objeto da mesma forma que as partes foram carregadas. Se um objeto foi originalmente carregado como

um único objeto, o Gerenciador de transferência do S3 v1 não é capaz de acelerar o processo de download usando subsolicitações.

Comportamento com falha

Com o Gerenciador de transferência do S3 v1, uma solicitação de transferência de diretório falha se alguma subsolicitação falhar. Diferentemente da v1, o futuro retornado do Gerenciador de transferência do S3 v2 é concluído com êxito mesmo se algumas subsolicitações falharem.

Como resultado, você deve verificar se há erros na resposta usando o método

[CompletedDirectoryDownload.failedTransfers\(\)](#) ou o método

[CompletedDirectoryUpload.failedTransfers\(\)](#), mesmo quando o futuro for concluído com êxito.

Alterações no utilitário de metadados do EC2 da versão 1 para a versão 2

Este tópico detalha as alterações no utilitário de metadados Amazon Elastic Compute Cloud (EC2) do SDK para Java da versão 1 (v1) para a versão 2 (v2).

Alterações de alto nível

Alteração	v1	v2
Dependências do Maven	<pre><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt></pre>

Alteração	v1	v2
	<pre data-bbox="609 212 933 682"></dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</groupId> <artifactId>aws-java-sdk-core</artifactId> </dependency> </dependencies></pre>	<pre data-bbox="1084 212 1412 877"><dependencies> <dependency> <groupId> software.amazon.aws sdk</groupId> <artifactId>imds</artifactId> </dependency> <dependency> <groupId> software.amazon.aws sdk</groupId> <artifactId>apache-client</artifactId> </dependency> </dependencies></pre>
Nome do pacote	com.amazonaws.util	software.amazon.aws.sdk.imds
Abordagem de instanciação	<p data-bbox="592 1073 974 1150">Use métodos de utilitário estático; sem instanciação:</p> <pre data-bbox="609 1213 966 1365">String localHostName = EC2MetadataUtils.getLocalHostName();</pre>	<p data-bbox="1068 1073 1458 1150">Use um método estático de fábrica:</p> <pre data-bbox="1084 1213 1396 1323">Ec2MetadataClient client = Ec2MetadataClient.create();</pre> <p data-bbox="1068 1388 1458 1465">Ou use uma abordagem de criador:</p> <pre data-bbox="1084 1528 1445 1759">Ec2MetadataClient client = Ec2MetadataClient.builder() .endpointMode(EndpointMode.IPV6) .build();</pre>

Alteração	v1	v2
Tipos de clientes	Somente métodos de utilitário síncronos: <code>EC2MetadataUtils</code>	Síncrono: <code>Ec2MetadataClient</code> Assíncrono: <code>Ec2MetadataAsyncClient</code>

¹ [Versão mais recente](#). ² [Versão mais recente](#).

³ Observe a declaração do módulo `apache-client` para v2. A V2 do utilitário de metadados do EC2 requer uma implementação da interface `SdkHttpClient` para o cliente de metadados síncronos ou da interface `SdkAsyncHttpClient` para o cliente de metadados assíncronos. A seção [???](#) mostra a lista de clientes HTTP que você pode usar.

Solicitar metadados

Na v1, você deve usar métodos estáticos que não aceitem parâmetros para solicitar metadados para um recurso do EC2. Por outro lado, é necessário especificar o caminho para o recurso do EC2 como um parâmetro na v2. A tabela a seguir mostra as diferentes abordagens.

v1	v2
<pre>String userMetaData = EC2MetadataUtils.getUserData();</pre>	<pre>Ec2MetadataClient client = Ec2MetadataClient.create(); Ec2MetadataResponse response = client.get("/latest/user-data"); String userMetaData = response.asString();</pre>

Consulte as [categorias de metadados da instância](#) para encontrar o caminho que você precisa fornecer para solicitar uma parte dos metadados.

Note

Ao usar um cliente de metadados de instância na v2, é necessário usar o mesmo cliente para todas as solicitações para recuperação de metadados.

Alteração de comportamento

Dados JSON

No EC2, o serviço de metadados de instância (IMDS) em execução local exibe alguns metadados como strings formatadas em JSON. Um exemplo são os metadados dinâmicos de um [documento de identidade de instância](#).

A API da v1 contém métodos separados para cada parte dos metadados de identidade da instância, enquanto a API da v2 exibe diretamente a string JSON. Para trabalhar com a string JSON, é possível usar a [API de documento](#) para analisar a resposta e navegar pela estrutura JSON.

A tabela a seguir compara como recuperar metadados de um documento de identidade de instância na v1 e na v2.

Caso de uso	v1	v2
Recuperar a região	<pre>InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String region = instanceInfo.getRegion();</pre>	<pre>Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String region = instanceInfo.asMap().get("region").asString();</pre>
Recupere o ID da instância	<pre>InstanceInfo instanceInfo =</pre>	<pre>Ec2MetadataResponse response =</pre>

Caso de uso	v1	v2
	<pre>EC2Metada taUtils.getInstanc eInfo(); String instanceId = instanceInfo.insta nceId;</pre>	<pre>client.get("/lates t/dynamic/instance- identity/document"); Document instanceInfo = response.asDocumen t(); String instanceId = instanceInfo.asMap ().get("instanceId ").asString();</pre>
Recupere o tipo da instância	<pre>InstanceInfo instanceI nfo = EC2Metada taUtils.getInstanc eInfo(); String instanceType = instanceInfo.insta nceType();</pre>	<pre>Ec2MetadataResponse response = client.get("/lates t/dynamic/instance- identity/document"); Document instanceInfo = response.asDocumen t(); String instanceType = instanceInfo.asMap ().get("instanceTy pe").asString();</pre>

Diferenças na resolução do endpoint

A tabela a seguir mostra os locais que o SDK confere para resolver o endpoint para o IMDS. Os locais são listados em ordem decrescente em termos de prioridade.

v1	v2
Propriedade do sistema: <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	Método de configuração do criador do cliente: <code>endpoint(...)</code>
Variável de ambiente: <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	Propriedade do sistema: <code>aws.ec2MetadataServiceEndpoint</code>

v1	v2
Valor padrão: <code>http://169.254.169.254</code>	Arquivo de configuração: <code>~.aws/config</code> com a configuração <code>ec2_metadata_service_endpoint</code>
	Valor associado ao <code>endpoint-mode</code> resolvido
	Valor padrão: <code>http://169.254.169.254</code>

Resolução de endpoint na v2

Quando você define explicitamente um endpoint usando o criador, esse valor de endpoint tem prioridade sobre todas as outras configurações. Quando o código a seguir é executado, a propriedade do sistema `aws.ec2MetadataServiceEndpoint` e a definição do arquivo de configuração `ec2_metadata_service_endpoint` serão ignoradas, se existirem.

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
    .build();
```

Modo de endpoint

Com a v2, é possível especificar um modo de endpoint para configurar o cliente de metadados para usar os valores de endpoint padrão para IPv4 ou IPv6. O modo de endpoint não está disponível para a v1. O valor padrão usado para IPv4 é `http://169.254.169.254` e `http://[fd00:ec2::254]` para IPv6.

A tabela a seguir mostra as diferentes maneiras pelas quais é possível definir o modo de endpoint em ordem decrescente em termos de prioridade.

		Possíveis valores
Método de configuração do criador do cliente: <code>endpointMode(...)</code>	<pre>Ec2MetadataClient client = Ec2Metada taClient</pre>	<code>EndpointMode.IPV4</code> , <code>EndpointMode.IPV6</code>

		Possíveis valores
	<pre>.builder() .endpointMode(EndpointMode.IPV4) .build();</pre>	
Propriedades do sistema	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4, IPv6 (o uso de maiúsculas ou minúsculas não importa)
Arquivo de configuração: ~.aws/config	Configuração da <code>ec2_metadata_service_endpoint</code>	IPv4, IPv6 (o uso de maiúsculas ou minúsculas não importa)
Não especificado nas formas anteriores	O IPv4 é usado	

Como o SDK resolve **endpoint** ou **endpoint-mode** na v2

1. O SDK usa o valor definido no código no criador do cliente e ignora todas as configurações externas. Como o SDK vai gerar uma exceção se `endpoint` e `endpointMode` forem chamados no criador do cliente, o SDK usará o valor do `endpoint` de qualquer método usado.
2. Se você não definir um valor no código, o SDK examinará a configuração externa: primeiro as propriedades do sistema e depois uma configuração no arquivo de configuração.
 - a. O SDK primeiro confere o valor de um `endpoint`. Se um valor for encontrado, ele será usado.
 - b. Se o SDK ainda não encontrou um valor, ele procurará as configurações do modo de `endpoint`.
3. Por fim, se o SDK não encontrar configurações externas e você não tiver configurado o cliente de metadados no código, o SDK usará o valor IPv4 de `http://169.254.169.254`.

IMDSv2

O Amazon EC2 define duas abordagens para acessar os metadados da instância:

- O serviço de metadados de instância versão 1 (IMDSv1): uma abordagem de solicitação/resposta
- Serviço de metadados de instância versão 2 (IMDSv2): uma abordagem orientada a sessões

A tabela a seguir compara como os SDKs do Java funcionam com o IMDS.

v1	v2
O IMDSv2 é usado por padrão	Sempre usa o IMDSv2
Tenta buscar um token de sessão para cada solicitação e voltará para o IMDSv1 se não conseguir receber um token de sessão.	Mantém um token de sessão em um cache interno que é reutilizado para várias solicitações.

O SDK para Java 2.x é compatível somente com o IMDSv2 e não retorna ao IMDSv1.

Diferenças de configuração

A tabela a seguir lista as diferentes opções de configuração.

Configuração	v1	v2
Repetições	Configuração não disponível	Configurável por meio do método do criador <code>retryPolicy(...)</code>
HTTP	Tempo limite de conexão configurável por meio da variável de ambiente <code>AWS_METADATA_SERVICE_TIMEOUT</code> . O padrão é 1 segundo.	Configuração disponível ao transmitir um cliente HTTP para o método do criador <code>httpClient(...)</code> . O tempo limite de conexão padrão para clientes HTTP é de 2 segundos.

Exemplo de configuração de HTTP v2

O exemplo a seguir mostra como configurar o cliente de metadados. Este exemplo configura o tempo limite de conexão e usa o cliente do Apache HTTP.

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();
```

```
Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();
```

Alterações na CloudFront pré-assinatura da Amazon da versão 1 para a versão 2

Este tópico detalha as mudanças na Amazon CloudFront da versão 1 (v1) para a versão 2 (v2).

Alterações de alto nível

Alteração	v1	v2
Dependências do Maven	<pre><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>cloudfront</art ifactId> </dependency></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>cloudfront</art ifactId> </dependency></pre>

Alteração	v1	v2
	<code></dependencies></code>	<code></dependencies></code>
Nome do pacote	<code>com.amazonaws.services.cloudfront</code>	<code>software.amazon.awssdk.services.cloudfront</code>
Nomes da classe	CloudFrontUrlSigner CloudFrontCookieSigner	CloudFrontUtilities SignedUrl CannedSignerRequest CustomSignerRequest

¹ [Versão mais recente.](#) ² [Versão mais recente.](#)

Mudanças na API

Comportamento	v1	v2
Crie uma solicitação predefinida	Os argumentos são passados diretamente para a API.	<pre>CannedSignerRequest cannedRequest = CannedSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(privateKey) .keyPairId(keyPairId) .expirationDate(expirationDate) .build();</pre>

Comportamento	v1	v2
Crie uma solicitação personalizada	Os argumentos são passados diretamente para a API.	<pre> CustomSignerRequest customRequest = CustomSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(keyFile) .keyPairId(keyPairId) .expirationDate(expirationDate) .activeDate(activeDate) .ipRange(ipRange) .build(); </pre>
Gere um URL assinado (predefinido)	<pre> String signedUrl = CloudFrontUrlSigner.getSignedURLWithCannedPolicy(resourceUrl, keyPairId, privateKey, expirationDate); </pre>	<pre> CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create(); SignedUrl signedUrl = cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest); String url = signedUrl.url(); </pre>

Comportamento	v1	v2
Gere um cookie assinado (personalizado)	<pre> CookiesForCustomPolicy cookies = CloudFrontCookieSi gner.getCookiesFor CustomPolicy(resourceUrl, privateKey, keyPairId , expirationDate, activeDate, ipRange); </pre>	<pre> CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilitie s.create(); CookiesForCustomPolicy cookies = cloudFrontUtilitie s.getCookiesForCus tomPolicy(customRe quest); </pre>

Cabeçalhos de cookies refatorados na v2

No Java v1, o Java SDK fornece cabeçalhos de cookies como um `Map.Entry<String, String>`

```

Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"

```

O SDK Java v2 fornece o cabeçalho inteiro como um único `String`

```

String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"

```

Alterações na análise de URIs do Amazon S3 da versão 1 para a versão 2

Este tópico detalha as mudanças na análise de URIs do Amazon S3 da versão 1 (v1) para a versão 2 (v2).

Alterações de alto nível

Para começar a analisar um URI do S3 na v1, você instancia um usando um `AmazonS3URI` construtor. Na v2, você chama `parseUri()` uma instância de `S3Utilities`, para retornar um `S3URI`.

Alteração	v1	v2
Dependências do Maven	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>s3</artifactId> </dependency> </dependencies> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>s3</artifactId> </dependency> </dependencies> </pre>
Nome do pacote	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
Nomes da classe	AmazonS3URI	S3URI

¹ [Versão mais recente.](#) ² [Versão mais recente.](#)

Mudanças na API

Comportamento	v1	v2
Analise um URI do S3.	<pre>URI uri = URI.create("https://s3.amazonaws.com"); AmazonS3Uri s3Uri = new AmazonS3URI(uri, false);</pre>	<pre>S3Client s3Client = S3Client.create(); S3Utilities s3Utilities = s3Client.utilities(); S3Uri s3Uri = s3Utilities.parseUri(uri);</pre>
Recupere o nome do bucket de um URI do S3.	<pre>String bucket = s3Uri.getBucket();</pre>	<pre>Optional<String> bucket = s3Uri.bucket();</pre>
Recupere a chave.	<pre>String key = s3Uri.getKey();</pre>	<pre>Optional<String> key = s3Uri.key();</pre>
Recupere a região.	<pre>String region = s3Uri.getRegion();</pre>	<pre>Optional<Region> region = s3Uri.region(); String region; if (s3Uri.region().isPresent()) { region = s3Uri.region().get().id(); }</pre>
Recupere se o URI do S3 é do estilo de caminho.	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>
Recupere o ID da versão.	<pre>String versionId = s3Uri.getVersionId();</pre>	<pre>Optional<String> versionId =</pre>

Comportamento	v1	v2
		<pre>s3Uri.firstMatchin gRawQueryParameter ("versionId");</pre>
Recupere os parâmetros da consulta.	N/D	<pre>Map<String, List<Stri ng>> queryParams = s3Uri.rawQueryPara meters();</pre>

Alteração de comportamento

Codificação de URL

v1 fornece a opção de passar um sinalizador para especificar se o URI deve ser codificado em URL. O valor padrão é `true`.

Na v2, a codificação de URL não é suportada. Se você trabalha com chaves de objeto ou parâmetros de consulta que tenham caracteres reservados ou não seguros, você deve codificá-los em URL. Por exemplo, você precisa substituir um espaço em branco " " por. `%20`

Use o SDK para Java 1.x e 2.x lado a lado

Você pode usar as duas versões do AWS SDK for Java em seus projetos.

A seguir, um exemplo do arquivo `pom.xml` para um projeto que usa o Amazon S3 da versão 1.x e o DynamoDB da versão 2.16.1.

Exemplo Exemplo de POM

Este exemplo mostra uma entrada de arquivo `pom.xml` para um projeto que usa ambas 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>
```

Chave OpenPGP para o AWS SDK for Java

Todos os artefatos Maven disponíveis publicamente para o AWS SDK for Java são assinados usando o padrão OpenPGP. A chave pública necessária para verificar a assinatura de um artefato está disponível na seção a seguir.

Chave atual

A tabela a seguir mostra as principais informações do OpenPGP para as versões atuais do SDK para Java 1x e do SDK para Java 2.x.

ID da chave	0xAC107B386692DADD
Tipo	RSA
Tamanho	4096/4096
Created	30/06/2016
Expires	08-10-2024
ID de usuário	SDKs e ferramentas da AWS <aws-dr-tools@amazon.com>
Impressão digital da chave	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

Para copiar esta chave pública OpenPGP do SDK para Java na área de transferência, selecione o ícone “Copiar” no canto superior direito.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmFbxdJgz1lD7wr1skQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSWsX2psgvdmeyUpW9ap01rThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRrtwt5ktPAA5bM9ZZaGKriej
kT2lPffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WP1nxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
```

```
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+Uk1gjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+ff+Xf0Cl6by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFEMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q
TbcY5SMnkIwFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNARpWb3dPMThL2xAY+fs60vXdB1Sk0tYJpDWpFgvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SjQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxq1kkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRf3KD
0Sn5CbmXpAcHJ1ZHRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbtnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VHwQsaW
jMrGj000MFzxGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
16iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvvcMXnduLtkBEX7TISMPW+n+0Ta63/z4YfFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

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 4 de março de 2024.

Alteração	Descrição	Data
the section called “Com segurança”	Adicione instruções para desativar o IMDSv1.	14 de março de 2024
the section called “step-by-step Instruções S”	Adicione instruções de step-by-step migração.	8 de março de 2024
Migrar para a versão 2	Atualize o tópico de migração.	14 de fevereiro de 2024
the section called “Configurar os clientes HTTP baseados em AWS CRT”	Adicione informações sobre o cliente HTTP síncrono AWS baseado em CRT.	5 de janeiro de 2024
the section called “Amazon Cognito Identity” e the section called “Provedor de identidade do Amazon Cognito”	Os exemplos do Amazon Cognito foram movidos para a seção Exemplos de código.	28 de dezembro de 2023
Usar recursos do SDK	O tópico sobre recursos do SDK foi reformulado.	11 de dezembro de 2023
Chave OpenPGP	Forneça a chave OpenPGP atual.	6 de dezembro de 2023
the section called “Alterações na serialização”	Descreva as diferenças de serialização entre a v1 e a v2 do SDK para Java.	5 de dezembro de 2023
the section called “Gerenciador de transferências do S3”	Adição de uma seção que detalha as alterações no S3	13 de novembro de 2023

Alteração	Descrição	Data
	Transfer Manager da versão 1 para a versão 2.	
the section called “Referência de anotações”	Adição de uma lista de anotações de classes de dados que podem ser usadas com o cliente aprimorado DynamoDB.	30 de outubro de 2023
???	Adição de informações sobre o status de migração de bibliotecas e utilitários do SDK para Java v1.x para v2.x	17 de outubro de 2023
???	Atualização do tópico de configuração do Gradle	17 de outubro de 2023
the section called “Ignorar atributos nulos de objetos aninhados”	Adição de informações sobre a anotação do cliente aprimorado DynamoDB <code>@DynamoDbIgnoreNulls</code> .	22 de setembro de 2023
the section called “Acesso entre regiões”	Adição de informações sobre acesso entre regiões a buckets do Amazon S3.	31 de agosto de 2023
the section called “Preservar objetos vazios”	Adição de seção que discute a anotação <code>@DynamoDb PreserveEmptyObject</code> .	25 de agosto de 2023
???	Atualização da seção do cliente do serviço.	15 de agosto de 2023

Alteração	Descrição	Data
the section called “Recomendações do cliente”	Desde a versão 0.23, o AWS CRT suporta sistemas operacionais baseados em musl, como o Alpine Linux. As recomendações do cliente HTTP agora refletem o suporte musl.	11 de agosto de 2023
the section called “Criar políticas do IAM”	Adição da seção da API do construtor de política do IAM	31 de julho de 2023
the section called “Conceitos básicos”	Correção de vários trechos na seção Introdução do tópico do cliente aprimorado do DynamoDB.	24 de julho de 2023
the section called “Suporte de proxy”	Adição de informações e exemplos de suporte de proxy HTTP para cada cliente HTTP.	2 de junho de 2023
Reorganização do índice	Promoção da seção Exemplos de código e das entradas Trabalhe com Serviços da AWS dos TOC de nível superior.	24 de maio de 2023
the section called “Adicionar dependência de registro”	Mostrar as dependências do Gradle na seção de registro.	23 de maio de 2023
the section called “Trabalhar com resultados paginados”	Atualização do tópico de paginação.	18 de maio de 2023
the section called “Configurar um projeto do Gradle”	Atualização da configuração do projeto Gradle.	3 de maio de 2023

Alteração	Descrição	Data
API do cliente aprimorado do DynamoDB	Lançamento do tópico reescrito da API do cliente aprimorado DynamoDB.	28 de abril de 2023
Atualização das instruções do tutorial de introdução	Modificação do arquétipo do Maven para incluir a opção CredentialsProvider; instruções modificadas adequadamente.	11 de abril de 2023
the section called “Recomendações do cliente”	Adição de 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 Práticas recomendadas de segurança no IAM .	14 de março de 2023
the section called “Recarregar credenciais de perfil”	Adição de uma seção sobre como recarregar as credenciais do perfil.	9 de fevereiro de 2023
the section called “Configurar os clientes HTTP baseados em AWS CRT”	Atualização do tópico para o lançamento GA.	8 de fevereiro de 2023
the section called “Trabalhar com metadados da instância do Amazon EC2”	Adição de um exemplo guiado do cliente Java SDK para o serviço de metadados da instância Amazon S3.	1º de fevereiro de 2023
the section called “Usar um cliente do S3 de alta performance”	Adicione uma seção para o cliente AWS S3 baseado em CRT.	19 de dezembro de 2022

Alteração	Descrição	Data
the section called “Transferir arquivos e diretórios”	Atualização dos exemplos do Gerenciador de transferências do Amazon S3 para a versão GA.	19 de dezembro de 2022
the section called “Melhores práticas”	Adição da seção de melhores práticas.	18 de novembro de 2022
the section called “Carregar credenciais temporárias de um processo externo”	Adição de uma seção sobre o carregamento de credenciais de um processo externo.	15 de novembro de 2022
the section called “Métricas de cliente de serviço”	Atualização da lista de métricas com o requisito de uso do cliente HTTP.	9 de novembro de 2022
the section called “Transferir arquivos e diretórios”	Correção do código de exemplo.	2 de novembro de 2022
the section called “Reduza o tempo de inicialização do SDK para AWS Lambda”	Atualização da seção com opções adicionais para reduzir o tempo de inicialização do Lambda.	1º de novembro de 2022
the section called “Clientes HTTP”	Adição de informações de configuração para cobrir todos os clientes HTTP no SDK.	26 de outubro de 2022
the section called “Registro em log”	Atualização do tópico de registro para incluir detalhes de registro de conexão para todos os clientes HTTP.	04 de outubro de 2022
the section called “serviços de banco de dados AWS”	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

Alteração	Descrição	Data
O EC2-Classic Networking está será descontinuado	O EC2-Classic está será descontinuado em 15 de agosto de 2022.	28 de julho de 2022
the section called “Opções de autenticação adicionais”	A atualização da dependência é necessária para autenticação única.	18 de julho de 2022
the section called “Transport Layer Security (TLS)”	Atualização das informações de segurança do TLS.	8 de abril de 2022
the section called “Opções de autenticação adicionais”	Adição de mais informações sobre configuração e uso de credenciais.	22 de fevereiro de 2021
the section called “Configurar um projeto do GraalVM Native Image”	Novo tópico para configurar um projeto GraalVM Native Image.	18 de fevereiro de 2021
the section called “Pesquisar estados de recursos”	Waiters lançados; tópico adicionado para o novo atributo.	30 de setembro de 2020
the section called “Usar métricas do SDK”	Métricas lançadas; tópico adicionado para o novo atributo.	17 de agosto de 2020
the section called “Amazon SNS”	Foram adicionados exemplos de tópicos para Amazon SNS.	30 de maio de 2020
the section called “Reduza o tempo de inicialização do SDK para AWS Lambda”	Tópico de desempenho da AWS Lambda função adicionado.	29 de maio de 2020
the section called “Definir o JVM TTL para pesquisas de nome DNS”	Adição do tópico de cache de DNS do JVM TTL.	27 de abril de 2020

Alteração	Descrição	Data
the section called “Configurar um projeto do Apache Maven” , the section called “Configurar um projeto do Gradle”	Novos tópicos de configuração do Maven e do Gradle.	21 de abril de 2020
the section called “Transport Layer Security (TLS)”	O TLS 1.2 foi adicionado à seção de segurança.	19 de março de 2020
the section called “Inscrever-se no Amazon Kinesis Data Streams”	Exemplos de Kinesis stream adicionados.	2 de agosto de 2018
the section called “Trabalhar com resultados paginados”	Adição do tópico de paginação automática.	5 de abril de 2018
???	Foram adicionados exemplos de tópicos para IAM Amazon EC2, CloudWatch DynamoDB e.	29 de dezembro de 2017
the section called “Amazon S3”	Exemplo de getObject adicionado para Amazon S3.	7 de agosto de 2017
the section called “Usar programação assíncrona”	Adição do tópico assíncrono.	4 de agosto de 2017
Versão GA do AWS SDK for Java 2.x	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.